Integració de tecnologies paral·leles per al processament de treballs biomèdics.

Autor: Gabriel Aparício i Pla
Directors: Ignacio Blanquer Espert
Vicente Hernández García

Ciutat de València, 1 de setembre de 2008

AGRAÏMENTS

Abans d'encetar un camí cal nodrir-se dels recursos suficients per fer front a les inclemències, sovint imprevisibles. Començar una aventura com és el treball científic en el marc d'un màster oficial de doctorat no és un camí a priori senzill, i molt més si es tracta d'una primera promoció, com és el cas. Els agraïments van, per tant, en eixa direcció. En primer lloc, les gràcies als directors, Nacho Blanquer i Vicente Hernández, per haver dipositat en mi la seua confiança, haver-me ensenyat on començava el camí i contribuir amb la seua dedicació a anar traçant-lo.

Pel camí m'he anat trobat amb molta gent que ha ajudat a enriquir el viatge, que amb consells, noves motivacions i aportacions diverses, han fet del meu viatge una aventura molt més interessant i productiva. Bàsicament caldria parlar del doctor Raúl Isea, Stefan Goetz, Ana Conesa, Javier Tamames i Miguel Pignatelli.

Després estarien els meus companys de viatge, sobretot l'amic Damià, que tant de temps i esforços ha invertit tant en la finalització del meu projecte final de carrera de l'enginyeria en informàtica, com en tots els treballs que conformen aquesta memòria de màster. També vull agrair la tasca dels administradors dels equips, Carlos, Miguel i José Vicente, per la paciència infinita i la seua col·laboració en tot moment per tindre els recursos del nostre grup en perfecte estat. No vull deixar passar tampoc el moment per mostrar la meua gratitud pels consells, solidaritat i comprensió a Andrés, Carlos, Cristina, Eli, Eloy, Erik, Gabi, Germán, Javier, José Vicente, Philippe, Roberto i Santi.

Entre els amics més íntims, deixe un breu apartat per a Josep, que és possiblement qui més clar tenia fins on podia arribar i no ha dubtat mai en deixar el seu camí per fer-me una visita. Per últim, el meu reconeixement va dedicat als meus pares Luis i Rosa pel seu suport incondicional, als meus germans Cèsar i Maite i al meu cosí Juan Carlos per haver obert sempre el camí i deixar-me veure els mapes fruit de les seues exploracions i la meua dona Laura per mai defallir ni quan m'ha costat trobar la brúixola.

A tots ells, i als que em deixe a propòsit, moltes gràcies.

ÍNDEX DE CONTINGUTS

ÍNDEX DE FIGURES

ÍNDEX D'EQUACIONS

ÍNDEX DE TAULES

1. INTRODUCCIÓ

Després de la reforma educativa impulsada pels acords recollits a l'anomenada *Declaració de Bolonya* [1] del 1999, a la *Universitat Politècnica de València* al 2006 naixen els primers *Màsters Oficials de Doctorat* i, entre ells, el *Màster en Computació Paral·lela i Distribuïda*. Dintre del citat màster hi ha, en forma d'una assignatura de 20 crèdits ECTS, la *Tesi del màster*, la memòria de la qual és justament aquest document.

A la present memòria s'intenta compactar en un sol document, els treballs i esforços realitzats per l'autor a l'hora de fusionar tres enfocaments de computació: la computació paral·lela, la computació *Grid* [2] i la computació d'altes prestacions, en el marc de la bioinformàtica. Al llarg de nou capítols, es pretén fer un recorregut lògic, a mode de crònica d'un viatge encara inacabat, on l'autor intenta ordenar i sintetitzar el seguit d'experiències per les que ha anat passant, remarcant amb exemples pràctics el profit de tots els coneixements teòrics assolits.

La memòria consta de sis capítols, incloent la present introducció que cobreix el primer d'ells. El segon capítol farà un repàs a l'estat de l'art dels principals estàndards, paradigmes i eines que seran la base sobre la que es recolzarà la resta de treball desenvolupat. Al tercer capítol hi ha una anàlisi de la integració actual de l'estàndard de programació distribuïda MPI [3],[4] dintre de la infraestructura *Grid* dels projectes EGEE [5] i EELA [6].

Els treballs pràctics desenvolupats estan distribuïts entre els capítols quart i cinqué. En el quart capítol es parla sobre una proposta de desplegament bastant genèric a una infraestructura *Grid*, com la d'EGEE, fent un repàs a tots els requeriments i necessitats d'una sèrie d'experiments pràctics, com els que es portaran a terme al següent capítol. Al cinqué capítol hi haurà tot un seguit d'experiències pràctiques amb dos conjunts de dades reals (els metagenomes del mar dels sargassos i tretze metagenomes de microbiota intestinal humana publicats recentment) i un conjunt de dades sintètic. Amb aquestes dades es pretén testejar distintes estratègies per abordar problemes concrets amb distints nivells de paral·lelisme integrats. Hi haurà experiments amb un nivell de paral·lelisme (nivell *Grid*), on a partir d'experiments seqüencials es valoraran l'efecte de vàries estratègies de particionament de dades o, distribució de càrrega; amb dos nivells de paral·lelisme (nivell *Grid* i nivell MPI), on es faran comparatives de versions seqüencials i paral·leles de memòria distribuïda d'un mateix algorisme; i amb tres nivells de paral·lelisme (nivell *Grid*, nivell MPI i nivell fils d'execució), on es contrastarà l'impacte d'aquests tres nivells en les prestacions d'una sèrie d'aplicacions desenvolupades adhoc.

Al sisé capítol es recolliran les principals conclusions extretes del treball realitzat, les publicacions a les quals ha donat lloc el treball generat (un total de 13 publicacions en forma de presentacions, ponències en congressos, publicacions tècniques o portals web) i un recull de les línies

principals per les quals hauria de continuar el present treball en un futur immediat. Finalment, es donarà pas a un conjunt de referències imprescindibles per entendre certes parts de la memòria, i que han aprofitat de base sobre la qual s'ha anat construint la tasca ací descrita.

2. ESTAT DE L'ART

Com a part essencial de qualsevol treball d'investigació, està l'anàlisi de les eines disponibles en l'actualitat relacionades amb el present estudi. Primerament es parlarà sobre les eines de computació paral·lela, des de la vessant de memòria distribuïda i la de memòria compartida. Després s'analitzarà la tecnologia *Grid*, fent èmfasi en alguns dels *middlewares* existents més emprats. Més avant es farà un repàs d'algunes aplicacions i algorismes relacionats amb la biomedicina, que aprofitaran de base computacional per als experiments sobre casos d'estudi reals. Finalment, es farà un comentari detallat dels dos grans casos d'estudis utilitzats als tests.

2.1. Computació Paral·lela

Un dels primers aspectes a analitzar de l'estat de l'art serà la part referent a la computació paral·lela en la seua visió més genèrica. És a dir, des dels paradigmes de memòria distribuïda, representats per MPI, fins als paradigmes de memòria compartida, representats pels fils de POSIX [7] i l'estàndard OpenMP [8].

2.1.1. Memòria Distribuïda: MPI

MPI es l'estàndard "de facto" en la programació dirigida a entorns paral·lels amb memòria distribuïda, MPI és un estàndard que defineix principalment un model de comunicacions independent de les plataformes i llenguatges emprats en entorns de computació paral·lela. Les principals finalitats de MPI són les altes prestacions, la indepenència de la topologia de interconnexió, l'escalabilitat i la portabilitat. Mentre que generalment es considera que ha aconseguit de forma satisfactòria totes aquestes metes, també ha estat criticat per ser de massa baix nivell i difícil d'emprar. La versió MPI-2 de l'estàndard pretén ampliar la funcionalitat cobrint la tolerància a errades, l'accés paral·lel al disc i altres característiques més avançades.

La majoria d'implementacions MPI consisteixen en un conjunt específic de rutines (API) que es pot cridar des de *Fortran* [9], *C* [10] o *C++* [11] i des de qualsevol altre llenguatge que siga capaç de gastar les citades llibreries, encara que hi ha implementacions particulars incompletes per a molts altres llenguatges. Els avantatges de MPI sobre les antigues llibreries de pas de missatges és la portabilitat (per què MPI ha estat implementat per a pràcticament totes les arquitectures de memòria distribuïda) i la velocitat (perquè hi ha implementacions optimitzades per a recursos específics). MPI es pot emprar en arquitectures de memòria compartida i NUMA (*Non-Uniform Memory Access*), on aporta la seua portabilitat i també ajuda a aconseguir altes prestacions en aplicacions orientades a la computació intensiva específiques per a l'arquitectura

particular, encara que les prestacions siguen en general prou reduïdes en aquest tipus d'arquitectures.

MPI és una especificació, no una implementació. MPI té especificacions independents de llenguatge per a les crides a funció i interfícies de les llibreries. El primer estàndard MPI especificava tant les interfícies per a *ANSI C* i *Fortran-77* com el LIS (Especificació Independent de Llenguatge). L'esborrany d'aquest estàndard va ser presentat a *Supercomputing 1994* [12] i finalitzat poc després. L'estàndard MPI-1.2 està comprés per al voltant de 128 funcions.

Hi ha dues versions de l'estàndard que són actualment populars: la versió 1.2 (què emfatitza el pas de missatges i té un entorn d'execució estàtic) i la versió MPI-2.1 (què inclou noves característiques com ara escalabilitat en l'entrada/eixida a fitxers, control de processos dinàmic i comunicació col·lectiva amb dos grups de processos). El LIS de MPI-2 especifica més de 500 funcions i ofereix interfícies per a *ANSI C*, *ANSI Fortran* (*Fortran90*) i *ANSI C++*. La inter-operabilitat d'objectes definits en MPI també va estar afegida per tal de permetre amb més facilitat la programació amb pas de missatges barrejant llenguatges de programació. Un efecte secundari de l'estandardització de MPI-2 (finalitzada en 1996) va ser la clarificació del estàndard MPI-1, creant el nivell MPI-1.2.

Convé notar que els programes MPI-1.2 (ara anomenats programes de llegat MPI-1) encara funcionen sota l'estàndard MPI-2, encara que algunes funcions estan ara considerades com a desfasades. Açò és important, ja que així queda assegurada la compatibilitat amb els programes més antics, que només usen el subconjunt de MPI-1.

Moltes vegades es compara MPI amb PVM [13], el qual fou un sistema molt popular per a entorns distribuïts i amb pas de missatges desenvolupat en 1989 i posà de relleu la necessitat d'aconseguir un estàndard per als sistemes paral·lels de pas de missatges.

2.1.1.1. Funcionalitat

La interfície de MPI proporciona funcionalitat de topologia virtual essencial, sincronització i comunicació entre un conjunt de processos (que ha estat mapejat a nodes) d'una manera independent del llenguatge, mitjançant els anomenats *language bindings*, més unes quantes característiques que són dependents del llenguatge. Els programes de MPI sempre funcionen amb processos, encara que és comú trobar a la bibliografia com de vegades es confon el concepte procés amb el concepte processador. A l'hora d'aconseguir les màximes prestacions, es tria un procés per processador (o més recentment, per *core*). Açò forma part del mapeig, que ocorre en temps d'execució a través de l'agent que inicia el programa MPI, normalment anomenat *mpirun* o *mpiexec*.

Tals funcions inclouen (però no es limiten exclusivament) operacions del tipus punt a punt d'enviament i recepció, triant entre topologies lògiques de procés del tipus cartesià o graf, intercanviant les dades entre parells de processos (operacions d'enviament i recepció), combinant resultats

parcials de càlculs (operacions de reunió i reducció), sincronitzant nodes (operació de barrera) així com obtenint informació referent a la xarxa com ara el nombre de processos en la sessió, identitat actual del processador on un procés està mapejat, processos veïns accessibles en una topologia lògica, etc. Les operacions punt a punt estan disponibles en versions síncrones, asíncrones, amb *buffer* o *ready*, per tal d'aconseguir un control del nivell de sincronització per als enviaments. Moltes operacions excepcionals són possibles en el mode asíncron en la majoria d'implementacions.

El valor relatiu de solapar comunicació i còmput, les transferències sincròniques front a les asíncrones i la baixa latència front a la baixa sobrecàrrega de les comunicacions són algunes de les importants controvèrsies que tenen els usuaris i comunitats de desenvolupadors MPI, més encara considerant que els avanços recents en l'arquitectura *multicore* faran reviure tots aquestos debats. MPI-1 i MPI-2 permeten implementacions que intenten aprofitar el solapament de comunicació i còmput, però amb algunes dificultats en la pràctica. MPI també especifica interfícies segures amb l'ús de fils, les quals tenen estratègies de cohesió i acoblament que ajuden a evitar la manipulació d'estats ocults insegurs dins de la interfície. Com a tal, és relativament fàcil escriure codi MPI amb suport a múltiples fils amb connexions punt a punt, i moltes implementacions són compatibles amb un codi així. Les comunicacions col·lectives amb múltiples fils obtenen millors prestacions amb còpies múltiples dels comunicadors.

2.1.1.2. Implementacions

- Implementacions per a clusters clàssics i supercomputadores

  El llenguatge d'implementació per a MPI és diferent en general del llenguatge que es gasta en temps d'execució. La majoria d'implementacions MPI estan fetes en una combinació de *C*, *C++* i assemblador, i es gasta *C*, *C++* i *Fortran* a la programació. Tanmateix, el llenguatge d'implementació i el llenguatge emprat per l'usuari final estan en principi desacoblats.

  Una de les implementacions inicials més populars de l'estàndard MPI 1.x va ser MPICH, de *Argonne National Laboratory* i la *Mississippi State University*. IBM també va ser un dels primers en fer una implementació de l'estàndard MPI, i la majoria de companyies de supercomputadors al principi dels anys 1990 o bé gastaven MPICH o bé construïen les seues pròpies implementacions de l'estàndard MPI-1.x. LAM/MPI del *Ohio Supercomputing Center* va ser una altra de les primeres implementacions obertes. *Argonne National Laboratory* ha continuat desenvolupant MPICH durant més d'una dècada, i ara ofereix MPICH-2, el qual és una implementació de l'estàndard MPI-2.1. LAM/MPI s'ha juntat amb altres desenvolupadors MPI per a crear un altre projecte a nivell mundial anomenat OpenMPI, però aquest nom no implica cap connexió amb una versió de l'estàndard en concret. Hi ha molts altres esforços que són derivats de MPICH, LAM

i altres treballs, massa nombrosos per a nomenar-los Recentment, Microsoft ha inclòs una aportació MPI al seu *Cluster Computing Kit* (2005), basat en MPICH-2. MPI s'ha convertit i continua sent la interfície més utilitzada per a la programació concurrent en l'actualitat, i això ha donat lloc a intents per fer versions *Grid*, com ja es comentava a un article de Ian Foster i N.T. Karonis en el 1998 [14]. A partir d'eixa base, hi ha dues implementacions en forma de llibreries que s'imposen en l'actualitat, com són MPICH-G2 [15] i PACX-MPI [16], que es comentaran en capítols posteriors.

Moltes distribucions *Linux* inclouen MPI (MPICH i/o LAM com a exemples), però sempre és recomanable aconseguir versions noves dels llocs de desenvolupament MPI. La majoria dels fabricants tenen versions de codi obert especialitzades de MPICH, LAM i/o OpenMPI, les quals aporten millors prestacions i estabilitat.

A més de l'ús preponderant de MPI per a la programació enfocada a les altes prestacions, MPI també s'ha gastat àmpliament amb *Python*, *Perl* i *Java* [17], comunitats actualment en creixement. També han aparegut diverses versions d'un MPI compatible amb MATLAB, però no s'ha arribat al consens sobre una única forma de gastar MPI amb MATLAB. Les properes seccions detallaran un poc més aquestes aportacions.

- Python

Hi ha com a mínim cinc intents d'implementar MPI per a *Python*: *mpi4py*, *PyPar*, *PyMPI*, *MYMPI* i el submòdul MPI en *ScientificPython*. *PyMPI* és destacable pel fet que és una variant de l'intèrpret de *Python* convertint a l'aplicació multi-node l'intèrpret pròpiament dit, en lloc del codi que executa l'intèrpret. *PyMPI* implementa la majoria de l'especificació MPI i automàticament treballa amb el codi compilat que necessita fer crides MPI. *PyPar*, *MYMPI* i el mòdul de *ScientificPython* estan dissenyats per a treballar com un mòdul típic gastat sense més que una sentència import. Ells s'encarreguen de fer el treball del codificador, decidint quan i on pertany la crida a *MPI_Init*.

- OCaml

El mòdul *OCamlMPI* implementa un gran subconjunt de funcions MPI i s'empra actualment en computació científica. Per fer-se una idea de la maduresa del projecte, hi ha reportat en la *caml-list* més de 10.000 línies de codi en el programa *OCaml* relacionades amb MPI, gastant el citat mòdul, amb 500 línies addicionals i amb lleugeres reestructuracions del codi, obtenint excel·lents resultats, fins i tot amb 170 nodes en un supercomputador.

- Java

Encara que *Java* no té un enllaç MPI oficial, han hagut diversos intents de pontejar *Java* amb MPI, amb diferents graus d'èxit i compatibilitat. Un dels primers intents va ser el *mpiJava* de Bryan Carpenter, una col·lecció de *wrappers* JNI a llibreries locals de MPI en *C*, convertint-

se en una implementació híbrida amb una portabilitat limitada, el qual ha de ser recompilat amb la llibreria MPI específica que s'està gastant.

Tanmateix, aquest original projecte també definia la API de *mpiJava* (una API de MPI per a *Java* seguint els equivalents *C++*), la qual ha estat seguida per la resta de projectes MPI en *Java*. Una alternativa API, encara que menys emprada, és el MPJ API, dissenyada per ser més orientada a objectes i més propera a les convencions de codificació de *Sun Microsystems*. Altres APIs es basen en llibreries MPI en *Java* que són dependents d'una llibreria MPI local, o implementen les funcions de pas de missatges en *Java*, mentre que altres (com ara *P2P-MPI*) ofereixen funcionalitat *Peer-to-peer* i permeten operacions amb plataformes mixtes (per exemple, en *clusters* mixtes de *Linux* i *Windows*).

Moltes de les parts més complicades de qualsevol implementació MPI per a *Java* apareixen degut a les pròpies limitacions i peculiaritats del llenguatge, com ara la manca de punters i espai d'adreçament de memòria lineal per als seus objectes, que fa que la transferència de matrius multidimensionals i objectes complexos siga ineficient. Les solucions provisionals normalment implicaven la transferència d'una línia en un moment determinat i/o realitzar desserialització i *casting* tant a la part de l'enviament i a la de la recepció, simulant matrius de *C* o *Fortran* mitjançant l'ús de matrius unidimensionals i punters a tipus primitius amb l'ús de matrius amb un sol element, cosa que fa que l'estil de programació siga estrany comparant amb les convencions *Java*.

- Implementacions en maquinari

S'han realitzat esforços graduals en intentar implementar MPI directament al maquinari del sistema, per exemple mitjançant l'ús de la tècnica *Processor-in-memory* on les operacions MPI són construïdes dins de la microcircuiteria dels xips RAM en cada node. Aquest tipus d'implementació seria independent del llenguatge emprat, del sistema operatiu o del processador del sistema, però no és fàcil d'actualitzar.

S'ha realitzat una altra aproximació per tal d'accelerar el maquinari en una o més parts de l'operació. Aquesta inclouria processament per maquinari de les cues MPI o l'ús de RDMA (Accés Directe a Memòria Remot) per a transferir directament les dades entre la memòria i la interfície de xarxa sense necessitar la intervenció del processador o del nucli.

2.1.1.3. Adopció de MPI-2

Mentre que l'adopció de MPI-1.2 ha estat universal, estant present en pràcticament tots els *clusters* de computació, l'acceptació de MPI-2.1 ha estat més limitada. Aquestes podrien ser algunes de les principals raons.

Mentre que MPI-1.2 emfatitza el pas de missatges i un mínim entorn d'execució estàtic, les implementacions MPI-2 inclouen I/O i control de processos dinàmic i la grandària del *middleware* resultant és

considerablement més gran. A més, la majoria de llocs que gasten sistemes de planificació *batch* no poden donar suport a un control de processos dinàmic. La I/O paral·lela està amplament acceptada com un dels valors clau de MPI-2.

Gran part dels programes MPI-1.2 ja estaven desenvolupats abans d'aparéixer MPI-2 i funcionen bé. L'amenaça que suposava la pèrdua potencial de portabilitat per gastar funcions MPI-2 va fer la que la gent fora no s'atrevira a donar el pas durant molt de temps, encara que això ha canviat molt durant els últims anys i l'acceptació de MPI-2 és molt major.

Moltes aplicacions MPI-1.2 només gasten un subconjunt de l'estàndard (de 16 a 25 funcions). Aquesta minimalització de l'ús contrasta amb la gran quantitat de funcionalitat oferta per MPI-2.

Altres factors inhibidors poden ser citats, encara que normalment estan més relacionades en percepcions i creences que no en fets. Des de principis dels anys 2000, el suport donat a MPI-2 ha estat molt bo en implementacions lliures i comercials.

2.1.1.4. Futur de MPI

Hi ha diversitat de criteris respecte al futur de MPI. El *MPI Forum* ha estat inactiu durant aproximadament una dècada, però ha mantingut la seua llista de correu. Tanmateix, a finals de 2006, la llista de correu va ser reactivada per tal d'esclarir qüestions relacionades amb MPI-2, i possiblement per definir un nou nivell estàndard. El 9 de febrer de 2007 començava l'estàndard MPI-2.1, amb una nova *web* i una nova llista de correu. S'ha centrat inicialment en fomentar les discussions sobre errates, renovar els membres i l'interés i així explorar oportunitats futures.

La pervivència de MPI com a interfície està garantida als nivells MPI-1.2 i MPI-2.1 per molts anys. Igual que passa amb *Fortran*, està molt present en la computació tècnica, s'ensenya a totes bandes i es gasta a tot arreu. El fet que tant productes lliures com comercials facen ús de MPI assegura la seua continuïtat.

Les arquitectures estan canviant, amb una gran capacitat interna de concurrència (*multicore*), millor control concurrent de gra fi (*threading*) i més nivells de jerarquia de memòria. Açò ja ha provocat l'aparició de diversos estàndards complementaris per a la programació SMP, com per exemple OpenMP. Tanmateix, en el futur, tant la concurrència de gran escala i la multi-granular presenten serioses limitacions a l'estàndard MPI, el qual només és tangencialment amigable a la programació multifil i no s'especifica massa coses sobre com els programes multifil han de ser escrits. Mentre que hi ha implementacions MPI que donen suport multifil, el nombre d'aplicacions multifil amb pas de missatges són poques. Aconseguir concurrència multifil dins de MPI és tant un repte com una oportunitat per al futur estàndard.

El nombre de funcions és molt gran, com ja s'ha comentat, però el nombre de conceptes diferents és relativament baix. Tanmateix, tenint en compte que la majoria d'usuaris no fan ús ple de la capacitat de MPI-2, un

estàndard futur hauria de ser més compacte i també més específic o tindre diferents perfils per permetre a diferents tipus d'usuaris obtindre allò que necessiten sense esperar a un implementació completa o tindre tot el codi validat des del punt de vista d'enginyeria del programari.

La computació *Grid* ofereix una forma de control de processos estàtic i dinàmic particular per a MPI. Mentre que és possible forçar l'ús d'un model MPI en un *Grid*, la tolerància a errades i els programes de llarga durada suposen un problema actualment. Els *Grids* poden gastar el API de MPI per instanciar-lo entre conjunts de processos en execució, però es fa necessari un *middleware* multinivell que s'encarregue del control de la concurrència, les errades i el trànsit de missatges. Tant un MPI amb tolerància a errades com un MPI *Grid* han estat treballats però el model de programació definit, orientat a un gra fi, pot suposar un fre.

2.1.2. Memòria compartida: Fils de POSIX i OpenMP.

Pel que fa referència a les tendències en la programació dirigida a entorns paral·lels amb memòria compartida, s'analitzarà, per una banda, l'estàndard de fils de POSIX i després una API d'alt nivell que s'està imposant anomenada OpenMP.

2.1.2.1. Fils de POSIX

Els fils de POSIX són un estàndard de POSIX per a fils d'execució, on POSIX és l'acrònim de Interfície de Sistema Operatiu Portable basat en *Unix*. L'estàndard defineix una API per a crear i manipular fils d'execució. Les llibreries que implementen els fils de POSIX estàndard s'anomenen sovint *Pthreads*. Els *Pthreads* són més comunament gastats en sistemes POSIX, com ara *Linux* i *Solaris*, encara que hi ha implementacions per *Microsoft Windows*. Per exemple, els *Pthreads-w32* estan disponibles per a plataformes *Microsoft Windows* i donen suport a un subconjunt de l'API dels fils de POSIX.

L'interés en treballar amb múltiples fils d'execució ve alimentat per l'augment en les capacitats físiques dels nous processadors i per l'aparició de tecnologies com el *hyperthreading* o el *multicore*, sumats a l'abaratiment de les màquines amb diversos processadors. Per això, els fils de POSIX s'han convertit en una eina fonamental per traure un rendiment òptim als programes desenvolupats, especialment en aquells enfocats en l'obtenció d'unes altes prestacions.

Per altra banda, encara que el nivell de l'API dels fils de POSIX no es pot considerar baix, també hi ha qui opta per solucions de més alt nivell a costa de perdre un cert control del procés. Una de les solucions més comunament gastades en l'actualitat és OpenMP (que poden fer ús dels propis fils de POSIX) i s'analitzarà al següent apartat.

2.1.2.2. Introducció a OpenMP

OpenMP és una API que ens permet afegir concurrència a les aplicacions mitjançant paral·lelisme amb memòria compartida. Es basa en la creació de fils d'execució paral·lels compartint les variables del procés pare que els crea. Actualment està disponible en diverses plataformes i llenguatges, des de les derivades de *Unix* fins a la plataforma *Windows*, des d'un simple ordinador de sobretaula fins a un supercomputador i amb diverses extensions a llenguatges com *C*, *C++* o *Fortran*. Consisteix en un conjunt de directives de compilació, rutines de llibreria i variables d'entorn que influeixen en el comportament de l'execució. Gràcies al suport d'un grup amb importants empreses de fabricants de maquinari i programari, OpenMP constitueix un model portable i escalable que ofereix als programadors una interfície simple i flexible per a desenvolupar aplicacions paral·leles. És habitual vore l'ús de OpenMP en models híbrids per a computació paral·lela (normalment *clusters*) on es combina amb MPI, com queda constància en capítols posteriors de la present tesi.

A la Figura 1 es pot apreciar un exemple senzill d'execució paral·lela genèrica, per ilustrar el funcionament d'OpenMP. A la part de dalt veiem quina seria el treball global a realitzar, on hi ha tres tasques que s'han d'executar seqüencialment, on cadascuna d'elles pot ser dividida en distintes subtasques que es poden executar en paral·lel entre elles.

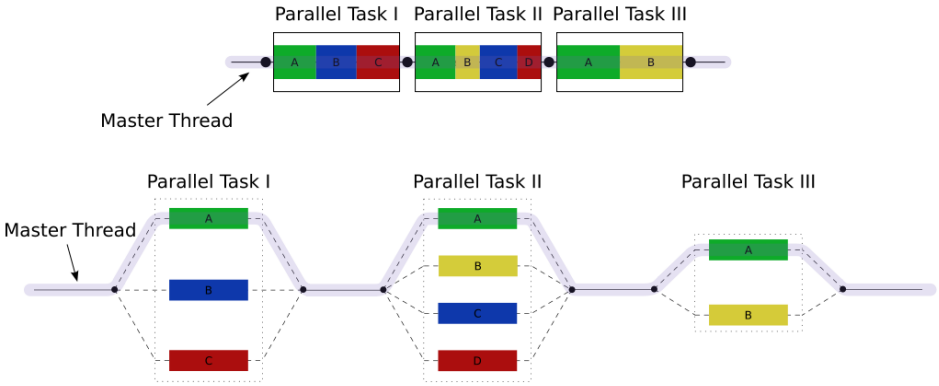

Figura 1: Traç d'una execució de vàries tasques paral·leles amb fils.

El treball d'OpenMP consistirà en crear els distints fils d'execució necessaris per aconseguir realitzar en paral·lel les tasques anteriorment citades. A la primera tasca crearia dos fils (a banda del fil mestre), a la segona tres i a la tercera només un. D'aqueixa forma s'ha reduït el cost temporal del programa a la suma dels costos de la subtasca més llarga de la tasca primera, la subtasca més llarga de la tasca segona i la subtasca més llarga de la tasca tercera.

2.1.2.3. Elements bàsics d'OpenMP

Els elements bàsics d'OpenMP es podrien resumir en l'esquema de la Figura 2, on veiem cinc grans branques de l'extensió del llenguatge OpenMP.

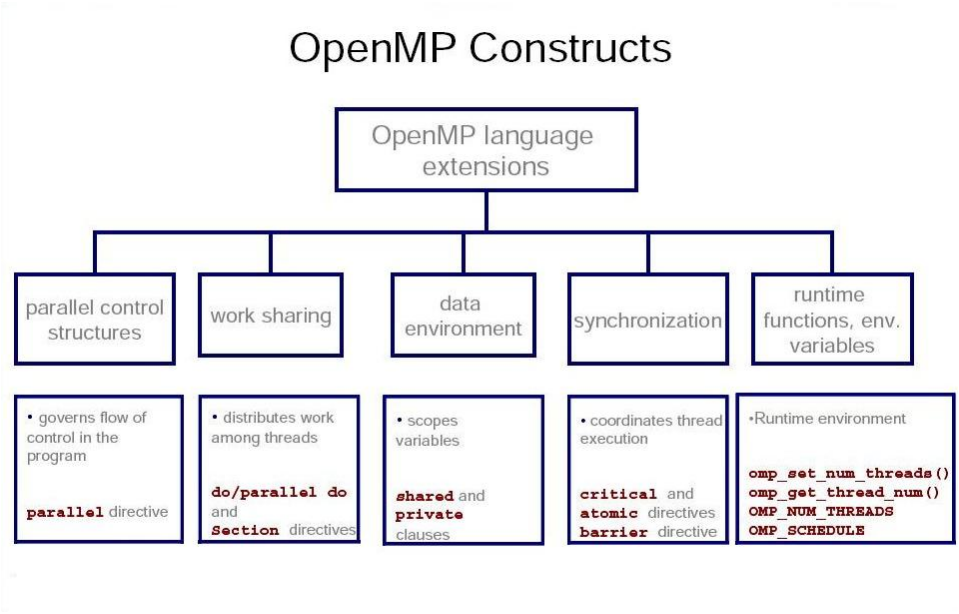

Figura 2: Extensions del llenguatge OpenMP.

Analitzant-les una per una tenim:

- Estructures de control paral·lel: s'encarreguen de monitoritzar el control de flux del programa (directiva *parallel*).
- Compartició de treball: distribueix el treball entre els distints fils d'execució (directives *do/parallel do* i *section*).
- Abast de les variables: s'indica quin és l'abast exacte del conjunt de variables definit en una secció concreta del codi (clàusules *shared* i *private*).
- Elements de sincronització: s'encarreguen de coordinar l'execució dels fils per tal de garantir la correcció dels programes quan hi ha conflictes en l'accés a dades (directives *critical*, *atomic* i *barrier*).
- Funcions i variables en temps d'execució: són capaces de modificar variables d'entorn en temps d'execució, com ara variar el nombre de fils d'execució o l'estratègia de planificació en l'assignació de subtasques als diferents fils (per exemple, la funció *omp_set_num_threads()*, o la variable d'entorn *OMP_NUM_THREADS*).

2.2. Tecnologia *Grid*

La definició de *Grid* es podria extraure de la donada per Ian Foster i Carl Kesselman [18]: "Un *Grid* computacional és una infraestructura de maquinari i programari que proveeix accés consistent a baix cost a recursos computacionals d'alt nivell". I es podria complementar amb la sentència visionària de Len Kleinrock en 1969: "Probablement vorem la distribució de les "utilitats de computació" tal i com es presenten l'electricitat i el telèfon, com serveis individuals en cases i oficines a través de tot el país".

La diferència entre la computació *Grid* i la computació paral·lela rau justament en la pròpia definició, on es remarca el paper dels diferents recursos que conformen una infraestructura *Grid* i el grau d'acoblament entre ells. Mentre que la computació paral·lela farà incís en recursos d'unes característiques similars (normalment processadors o nodes d'un *cluster* idèntics) i un alt grau d'acoblament entre els processos dels diferents recursos, a la computació *Grid* els recursos poden tindre papers molt diversos (recursos específics d'emmagatzematge, de càlcul, de gestió de càrrega, de catàleg, etc.) i el grau d'acoblament és lògicament molt baix. A més, els recursos d'una infraestructura *Grid* comprenen diversos dominis administratius, la qual cosa suposa un grau d'heterogeineïtat inherentment major que en el cas paral·lel.

En el present capítol es farà una revisió dels principals aspectes de la tecnologia *Grid* que convé tindre en compte. En primer lloc es revisarà el projecte europeu EGEE, que oferirà la infraestructura física on llançar els treballs *Grid*. En segon lloc, també caldrà esmentar els dos *middlewares Grid* en què s'han gastat en els treballs de la present memòria per interactuar amb la infraestructura EGEE: *LCG-2* [19] i *gLite* [20]. Tampoc es pot obviar la ferramenta *Globus Toolkit* [21] [22], la qual s'analitzarà en la seua versió 4.X, centrant l'anàlisi en les utilitats més interessants i que han sigut utilitzades en la confecció dels serveis *Grid* i sistemes automatitzats. Finalment es farà un repàs als serveis *Grid* de GT4.X, útils per a desenvolupar interfícies entre els clients i les infraestructures *Grid*.

2.2.1. Globus Toolkit

Globus Toolkit 4 (GT4) serà una eina clau en els desenvolupaments del treball present, ja que es gastaran directament moltes de les eines proporcionades com pel fet que altres *middlewares* també gastats, com *LCG-2* o *gLite*, fan ús de *Globus Toolkit*.

Globus Toolkit ® és resultat dels desenvolupaments de la "globus alliance", iniciada per I. Foster, C. Kesselman i S. Tuecke en el Laboratori Nacional d'Argonne, a finals dels 90. *Globus Toolkit* és un conjunt de ferramentes de codi obert fonamental que ens ofereix ponts cap a la tecnologia *Grid*, permetent a la gent compartir potència de còmput, bases de dades i altres ferramentes *online* amb mesures de seguretat que són accessibles des d'una mateixa corporació, institució o travessant fronteres sense un sacrifici de l'autonomia local. El *toolkit* inclou serveis programari i llibreries per a monitoritzar recursos, descobrir-los i controlar-los, a més de seguretat i administració de fitxers. A més de ser una part central dels projectes científics i d'enginyeria (que inverteixen aproximadament 500 milions de dòlars en tot el món en total), el *Globus Toolkit* és un substrat sobre el qual les companyies líders en informàtica estan construint productes *Grid* comercials significatius.

El *toolkit* inclou un conjunt de programari per a seguretat, infraestructures d'informació, administració de recursos i de dades, comunicació, detecció d'errades i portabilitat. Està empaquetat com un conjunt de components

que poden ser utilitzats bé independentment o bé alhora per desenvolupar aplicacions.

Cada organització té la seua pròpia forma de treballar, i la col·laboració entre múltiples organitzacions està limitada per la incompatibilitat de recursos, com ara arxius de dades, computadores i xarxes. El *Globus Toolkit* va ser concebut per a eliminar tots aquestos entrebancs i afavorir unes col·laboracions sense fissures. Els seus serveis principals, interfícies i protocols permetran als usuaris accedir remotament a recursos com si ells estigueren físicament a la cambra on està la màquina, mentre simultàniament preservaran el control local sobre aquells que estan gastant els recursos, a més de dir com els han de gastar i quan.

El *Globus Toolkit* ha crescut gràcies a una estratègia de codi obert similar a la del sistema operatiu *Linux*, i distinta dels intents propietaris de creació de programari de compartició de recursos. Tot açò afavoreix que hi haja una adopció més ràpida i que s'haja convertit en tota una gran innovació tècnica, ja que la comunitat de codi obert proveeix contínues millores al producte.

El context essencial en què està inspirat *Globus Toolkit* el podem trobar als papers "Anatomia del *Grid*" de Foster, Kesselman i Tuecke i "Fisiologia del *Grid*" de Foster, Kesselman, Nick i Tuecke.

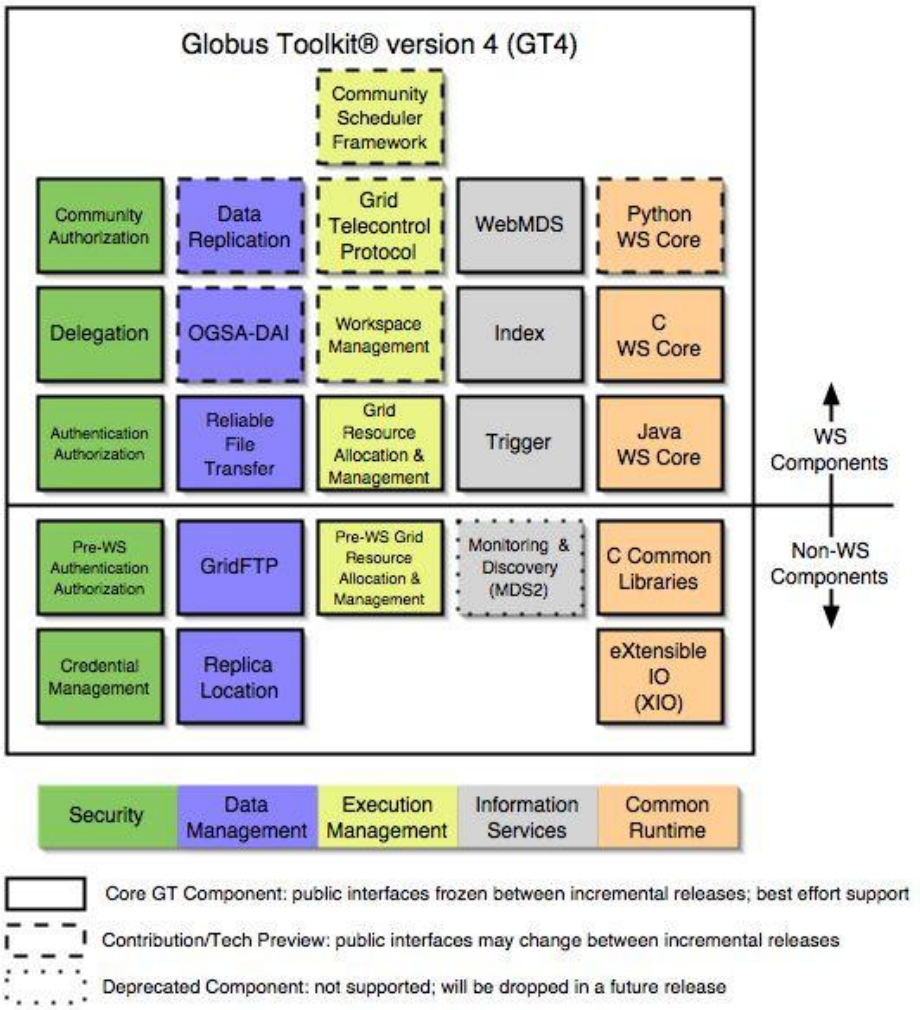

Figura 3: Esquema amb els principals components del Globus Toolkit ® versió 4 (GT4).

La presència dels Serveis Web (WS) a GT4 s'ha d'entendre en el fet que GT2 era un *middleware* orientat a entorns batch i era necessari sobrepassar les seues limitacions. Per això, gràcies a l'aparició del OGSA [23] en 2002 i després del fracàs del OGSI [24], s'opta per un desenvolupament basat en WS, com serà WSRF [25]. D'entre els diferents components que es poden vore a la Figura 3, veiem una clara divisió entre components WS i components no-WS. D'entre tots els components, es comentaran breument alguns d'ells, com ara la ferramenta GridFTP [26] [27], el MyProxy [28], els mecanismes d'autenticació i autorització, el control de credencials i altres components d'interés.

GridFTP és un protocol de transferència de dades d'altes prestacions, segur i confiable optimitzat per a xarxes amb grans amples de banda. El protocol GridFTP està basat en FTP, el més popular dels protocols de transferència de fitxers de Internet. En concret, s'han seleccionat un conjunt de característiques del protocol i altres extensions ja definides en els RFCs del IETF, també s'han afegit unes quantes característiques addicionals per poder satisfer els requeriments dels actuals projectes de *Grid* de dades.

MyProxy és un rebost de credencials *online*. És possible emmagatzemar credencials en el rebost MyProxy, protegides per una contrasenya, per a després recuperar-les en qualsevol lloc de la xarxa (MyProxy delegat). Açò elimina la necessitat d'haver de copiar manualment fitxers de claus privades i de certificats entre màquines. MyProxy pot, a més, ser usat per a autenticació als portals *Grid* i renovació automàtica de credencials amb administradors de treballs. Cal esmentar que des del punt de vista del nostre interés en el desenvolupament de serveis *Grid*, seran útils ambdues formes de funcionament. Per una banda, caldrà un MyProxy del qual es puga obtindre una delegació a l'hora de fer els llançaments dels treballs i per altra banda caldrà fer les renovacions de credencials pertinents ja que no es pot predir fàcilment la durada dels treballs i, per tant, la durada del *proxy*.

L'API d'autenticació està construït gastant les tecnologies d'infraestructura de clau pública (PKI), per exemple, certificats X.509 i TLS. A més de l'autenticació, hi ha una mecanisme de delegació basat en els certificats *proxy* X.509.

El suport per a l'autorització pren la forma d'un parell d'APIs. La primera dóna una API d'autorització genèrica que permet crides per a realitzar un control de l'accés basat en les credencials del client (és a dir, la cadena de certificat X.509). El segon ofereix una llista simple amb el control d'accés que relaciona les entitats remotes autoritzades amb els noms d'usuari del sistema local. El segon mecanisme a més aporta crides que permetran a tercers anul·lar el comportament per defecte i està actualment sent usat en el *Gatekeeper* i els servidors GridFTP.

Afegint al que ja s'ha comentat abans, hi ha vàries APIs i ferramentes de baix nivell per a administrar, descobrir i fer peticions de certificats. La part de GT 4.X dedicada als serveis *Web* gasta SOAP sobre HTTP per als missatges de comunicació. La seguretat en el nivell de missatge de l'autenticació i l'autorització en els serveis *Web* implementa l'estàndard

de seguretat dels serveis *Web* i l'especificació de converses segures per a serveis *Web* per tal de proveir els missatges SOAP d'una protecció dels missatges. Entre les característiques s'inclou l'autenticació del remitent, xifrat del missatge, protecció d'integritat del missatge i protecció contra repeticions.

La seguretat [29] en el nivell de transport de l'autenticació i autorització en els serveis *Web*, dóna un canal segur mitjançant l'ús de HTTP sobre SSL/TLS (HTTPS) per a transportar els missatges. Aquest mecanisme de seguretat casa amb totes les característiques de seguretat proveïdes per SSL/TLS amb l'addició del suport per als certificats *proxy* X.509.

El component *Authorization Framework* dóna un marc de treball per a l'autorització a nivell de contenidor. Açò permet cadenes de mòduls d'autorització amb interfícies ben definides per a ser associades amb diverses entitats, com per exemple, serveis en el contenidor. Això també fa possible distintes implementacions de diferents mòduls d'autorització, comprenent des del suport a l'autorització basada en *gridmap* fins a un mòdul que gasta el protocol SAML [30] per a fer demandes de serveis externs per a una decisió d'autorització.

El component *Web Services Grid Resource Allocation and Management* (WS GRAM) comprén un conjunt de serveis *Web* que obeeixen a WSRF per tal de localitzar, sotmetre, monitoritzar i cancel·lar treballs en els recursos de còmput *Grid*. WS GRAM no és un planificador de treballs, sinó simplement un conjunt de serveis i clients per a la comunicació amb un ventall de diferents planificadors de treballs de tipus *batch/cluster* gastant un mateix protocol. WS GRAM se suposa que ha d'adreçar un conjunt de treballs on les operacions fiables, la monitorització de l'estat, el control de les credencials i transferència de fitxers són molt importants.

2.2.2. EGEE i gLite

El projecte EGEE, actualment en la seua tercera fase, és la confluència de l'experiència de més de 150 centres de més de 27 països en un propòsit comú de contribuir als avanços recents de la tecnologia *Grid* i de desenvolupar un servei d'infraestructura *Grid* en producció que estiga disponible per als científics les 24 hores del dia.

La finalitat principal del projecte és proveir als investigadors de les universitats i de la indústria d'un accés a més recursos computacionals i d'emmagatzemament de dades, independentment de la seua localització geogràfica. El projecte EGEE també se centrarà en intentar atraure un nombre creixent de nous usuaris per al *Grid*.

El projecte es concentra fonamentalment en tres àrees centrals:

- La primera àrea és la de construir una infraestructura *Grid* consistent, robusta i segura, que puga atraure recursos computacionals addicionals.

- La segona àrea és la de comprovar, millorar i mantindre el *middleware* amb la finalitat de poder donar un servei eficient i confiable als usuaris.
- La tercera àrea seria la de promoure la recerca de nous usuaris i la consolidació dels actuals, tant de la indústria com del món científic, i assegurar que reben el procés d'aprenentatge i posterior suport que puguen necessitar, amb la implicació del desenvolupament de noves aplicacions i la promoció del seu ús.

El *Grid* EGEE aprofita la xarxa europea de recerca GÉANT i integra els coneixements generats per desenes de projectes *Grid* de la Unió Europea, nacionals i internacionals desenvolupats fins a la data actual.

Els centres proveïdors de recursos de EGEE es dividixen en 12 federacions, consistents en total en més de 240 centres, proveint de més de 41.000 CPUs.

El treball realitzat pel projecte està organitzat dins del total d'11 activitats. Dos dominis d'aplicació pilot van ser seleccionats per a guiar la implementació i certificar les prestacions i funcionalitats de la infraestructura resultant. Un és el *Large Hadron Collider Computing Grid* que dóna suport als experiments dels físics i un altre és el *Biomedical Grids*, on diverses comunitats estan afrontant reptes igualment encoratjadors per tal de respondre a l'allau de dades bioinformàtiques i del món de la salut. En l'actualitat, EGEE integra més de 5000 usuaris organitzats en 100 comunitats de 45 països.

Amb amb un finançament europeu superior a 16 milions d'euros anuals a través de la Comissió Europea, EGEE comença el seu quint any amb l'objectiu de trobar un model sostenible de finançament i amb el repte de ser la infraestructura computacional del Large Hadron Collider, que serà activat a finals del 2008.

gLite és la nova generació de *middleware* per a la computació *Grid*. Nascut des dels esforços col·laboratius de més de 80 persones en 12 diferents centres de recerca acadèmics i industrials com a part del projecte EGEE, *gLite* és un glite és un middleware orientat a la producció i que inclou una gran quantitat de serveis addicionals (com la tolerància a errades, la metaplanificació, els catàlegs de fitxers lògics, etc), que permetren construir aplicacions *Grid* avançades.

La primera versió de *gLite* oferia interessants ferramentes per a la planificació de la càrrega, el catàleg de fitxers en el *Grid* i una infraestructura de monitorització. Les versions futures afegeixen funcionalitats addicionals i components de re-enginyeria amb el propòsit de satisfer els requeriments dels principals dominis d'aplicació d'EGEE. Aquesta activitat *middleware* de EGEE treballa molt propera amb el projecte LCG.

Mentre el *middleware gLite* s'acabava d'enllestir, es realitzava una certificació per part de l'equip de desplegament, i s'ha instal·lat primer al servei de pre-producció, on es va testejat novament per grups d'aplicacions abans de ser més àmpliament distribuït. Els nous components estan ara mateix coexistint en el mateix sistema *Grid* amb el

middleware anterior (*LCG-2*), encetant una estratègia de desplegament progressiva per a dur a terme. Aquesta és una característica essencial per al nou *middleware* que ha de ser introduït dins d'un gran *Grid* operacional.

En maig de 2006 es va acabar d'enllestir la versió 3.0.0 de *gLite*. Per tal de minimitzar l'impacte en els sistemes actualment en producció i facilitar la transició des de 2.7, es va decidir realitzar una actualització per fases. La versió 3.0.0 pretén ser un híbrid entre *gLite* 1.5 i LCG 2.7, i la base per a la construcció d'un sol *middleware* on fer convergir a tota la comunitat EGEE.

Com ja s'ha comentat, hi ha diverses característiques de *gLite* que el fan interessant. Cal comentar que s'agrega una arquitectura orientada a serveis, major interoperabilitat amb altres rescursos, millor suport a catàlegs de dades, metadades i claus de xifrat. També l'aparició de nous gestors de càrrega orientats a serveis i noves interfícies a recursos. *gLite* intenta satisfer tots els requeriments i necessitats expressats per diverses comunitats d'usuaris durant el projecte EGEE.

Els principals components del paquet *middleware* són:

- El Globus Toolkit desenvolupat pel Globus Project.
- El sistema Condor desenvolupat a la Universitat de Wisconsin, Madison.
- Els components Globus i Condor i algunes altres ferramentes dels projectes dels EUA estan integrades i empaquetades com el Virtual Data Toolkit pel projecte VDT a la Universitat de Wisconsin, Madison. VDT dóna suport per a aquest paquet a LCG/EGEE.
- Ferramentes desenvolupades pel projecte DataGrid (EDG). El projecte fundat per la Unió Europea anomenat DataGrid va finalitzar en 2004, però les institucions que han desenvolupat les ferramentes requerides per al *Grid* LCG/EGEE van decidir continuar emprant-les fins que foren reemplaçades per programari millorat.
- Noves ferramentes i serveis com a conseqüencia de les anàlisis de requeriments fetes en les fases I i II del projecte EGEE.

El *middleware Grid gLite* comprén, entre d'altres, els següents elements:

- IS-BDII: *Information Service – Berkeley Database Information Index*. Aquest element proveeix la informació sobre els recursos *Grid* i els seus estats.
- CE: *Computing Element*; està definit com una cua de treballs *Grid*. Un element de còmput és una granja de nodes de còmput homogenis, anomenats *Worker Nodes*.
- WN: *Worker Node*; és una computadora encarregada d'executar els treballs.
- SE: *Storage Element*; un SE és un recurs d'emmagatzematge en el qual una tasca pot guardar dades que van a ser emprades per computadores del *Grid*.

- RC, RLS: *Replica Catalogue*, *Replica Location Service*; són els elements que controlen la localització de les dades *Grid*.
- RB: *Resource Broker*; el RB s'encarrega del balanceig de la càrrega dels treballs en el *Grid*, decidint en quins CEs seran llançats els treballs.
- UI: *User Interface*; aquest component és el punt d'entrada dels usuaris al *Grid* i proporciona un conjunt de comandaments i APIs que poden ser emprades pels programes per realitzar diferents accions en el *Grid*.
- *The Job Description Language* [31] (JDL); és la forma en la qual els treballs són descrits. Un JDL és un fitxer de text que especifica l'executable, els paràmetres del programa, els fitxers involucrats en el procés (safata d'entrada o *input sandbox* per als fitxers que es pujaran al RB i safata d'eixida o *output sandbox* per al conjunt de fitxers que s'han d'estar presents al RB en el moment de la descàrrega de resultats en el moment de finalització del treball) i altres requeriments addicionals.

2.2.3. Serveis Grid

A l'hora de buscar una interfície per als serveis desenvolupats, s'ha optat pel format de Servei *Grid*, per la seua portabilitat i compatibilitat, entre d'altres raons que caracteritzen aquesta arquitectura.

En l'actualitat, moltes de les aplicacions orientades a la xarxa estan fent ús dels Serveis *Web*, o *Web Services*. Aquesta tecnologia està àmpliament acceptada perquè, a més de suposar una regularització de la invocació a funcions remotes, ha vingut acompanyada d'entorns de desenvolupament amigables i marcs d'execució que han facilitat tant la implementació d'aquests serveis com el seu desplegament.

No obstant, el marc d'execució dels sistemes operatius no contempla conceptes com l'estat, el temps de vida, les notificacions, etc. És a dir, únicament es permet la creació de serveis estàtics i sense memòria. Aquest supòsit va en contra els models i llenguatges de programació actuals, en els quals es té una noció d'objecte, que pot ser creat i evolucionar en el temps, per a acabar sent destruït.

Els esforços dels programadors de sistemes operatius han estat dirigits en molts casos cap a l'oferiment de serveis en els quals, els missatges enviats tingueren implícit el nom o identificador d'un objecte sobre el qual es volguera actuar. En la part del servidor es té en funcionament un entorn d'execució orientat a objectes, que crearà, indexarà, recuperarà, i interactuarà amb objectes als què s'assigna l'identificador esmentat. No obstant, aquesta és una forma artificial de desenvolupar, que pot donar lloc a errors i inconsistències, ja que s'està tractant amb distints marcs d'execució per a un únic desenvolupament i una mateixa funcionalitat. Alguns exemples d'aquesta arquitectura pot ser WS que criden un servidor CORBA, components EJB exposades com WS, etc.

Addicionalment, ja que són un entorn estàtic, els sistemes operatius no contemplen (ni poden contemplar de forma natural) la idea d'esdeveniments en el sistema. Aquest fet suposa una tara important en el desenvolupament d'aplicacions convencionals ja que fins i tot des dels llenguatges de modelatge d'objectes (com l'UML), contemplen aquest concepte. Els Serveis *Grid* (SG), o *Grid Services*, plantegen solucions a algunes de les carències que s'identifiquen per als WS i per a això, ofereixen el concepte de Recurs en el marc dels sistemes operatius (*WS-Resource*). En el marc dels *WS-Resources*, es permet la declaració, creació, accés, monitorització i destrucció de recursos per mitjà de mecanismes convencionals de WS. D'aquesta manera, la tecnologia *Grid* complementa als sistemes operatius per a oferir serveis que tenen un estat, una concepció sobre el temps de vida i un sistema de notificació davant esdeveniments. Addicionalment, aquests GS vénen gestionats per un identificador que pot ser renovat al llarg del seu temps de vida.

Algunes d'aquestes característiques són molt importants en el disseny de les aplicacions actuals; no sols és important la característica de notificació d'esdeveniments i fallades o la gestió de noms i referències renovables, sinó que resulta essencial la concepció de l'estat i el temps de vida per als sistemes operatius. Aquesta característica serà la que habilite de forma natural les transaccions, que poden garantir execucions atòmiques sobre els sistemes d'informació, així com aplicacions que puguen ser executades de forma diferida.

Els GS es beneficien també de característiques addicionals de la tecnologia *Grid*, com és la seguretat GSI i la delegació de permisos, a transferència de fitxers via GridFTP, etc.

En definitiva, la definició del marc de *WS-Resources* facilita la construcció i utilització de serveis inter-operables, permetent que es puga crear, accedir i gestionar recursos que mantenen l'estat des de distints punts, en una forma estàndard. Al mateix temps, aquest marc ofereix suport per a la creació de recursos que mantenen l'estat, sense comprometre les característiques dels WS de ser implementats com a processadors de missatges sense més. En aquest marc de treball també es defineixen aspectes com la renovació de referències i la notificació d'esdeveniments i fallades.

2.3. Mètodes i aplicacions bioinformàtiques

En la següent secció, es farà un breu repàs a les principals aplicacions i algorismes emprats en els experiments emmarcats en el treball de la present tesi. Hi ha un grup més enfocat a la mineria de dades, on s'analitza el mètode dels K veïns més pròxims i el paquet WEKA que es gastarà en els experiments on s'integren els tres paradigmes de paral·lelisme. Aquestes aplicacions són d'especial interés per a la classificació en sistemes d'assistència a professionals, per exemple, en l'anàlisi d'imatges mèdiques o assistència en el diagnòstic de tumors a través de les característiques dels tumors. La raó d'escollir en concret el mètode dels K veïns més pròxims és el fet que és un mètode implementable de forma

molt senzilla, i així desenvolupar l'algorisme ràpidament (cosa que no és l'objecte del nostre estudi) i centrar-nos en l'anàlisi de com s'adapten les diverses versions desenvolupades a una infraestructura *Grid* real i així extraure conclusions pràctiques.

A més de la part de mineria de dades, també hi ha tres famílies d'aplicacions, amb les respectives versions seqüencials i paral·leles MPI, que treballen l'alineament de seqüències de proteïnes o nucleòtids. L'interés d'aquestes aplicacions és el d'avaluar des del punt de vista d'aplicacions *heretades*, bé per evitar el seu desenvolupament concret o per requeriments del propi client. Aquesta avaluació, a diferència de l'anterior, es farà només mitjançant combinació de MPI i *Grid* o exclusivament amb llançaments seqüencials a una infraestructura *Grid*. En concret es tracta de possiblement les tres més emprades en el camp de l'anàlisi genòmic i són BLAST, Mr.Bayes i CLUSTAL.

2.3.1. K veïns més pròxims (KNN)

Una de les tècniques de classificació basades en instàncies més populars és el mètode dels K veïns més pròxims (KNN) [32]. Aquest mètode és una generalització de la regla del veí més pròxim. Aquesta tècnica és apropiada quan hi ha una gran quantitat de registres o instàncies etiquetades i un grup menut de registres sense etiquetar que es pretén etiquetar amb la seua etiqueta més probable. La hipòtesi de partida és existeix una forta relació entre la distància multidimensional de cada entrada amb el seu etiquetatge. D'aquesta manera, el 1-NN (el veí més pròxim) consisteix en assignar als registres sense etiquetar, l'etiqueta del registre més proper en quant a distància multidimensional. Basats en açò, Cover & Hart implementaren una variant coneguda com a K-NN (K veïns més pròxims) que aplica la mateixa filosofia que el 1-NN però considerant l'etiqueta més freqüent en els K veïns més pròxims i no només amb el més proper. El paràmetre K és molt important i el seu valor òptim dependrà de molts factors, com ara la naturalesa de les dades, i ha de ser trobat experimentalment.

El test de la validació creuada és la tècnica de test més popular per al càlcul de l'error en la classificació. El procés comença amb un conjunt de registres etiquetats, els quals es divideixen en B blocs. Un dels blocs serà gastat com a conjunt de test i la resta com a conjunt d'entrenament. Aquesta operació serà realitzada amb els B blocs. El classificador K-NN és aplicat a tots els registres del bloc de test i es compara l'etiqueta assignada amb la seua etiqueta real. En cas que l'etiqueta siga diferent, es registra un error, i la suma de tots els errors obtinguts per a tot el bloc de test és l'error de validació. Aleshores caldrà repetir aquest procés per a cada bloc i la suma de tots els errors serà una aproximació al error real del classificador K-NN amb el paràmetre K elegit.

$$err = \sum_{b=1}^{b=B} err_b \text{ on b és el bloc elegit com a conjunt de test}$$

Equació 1: Càlcul de l'error real del classificador K-NN.

2.3.2. WEKA

WEKA [33] són les sigles de *Waikato Environment to Knowledge Analysis* (Entorn per a l'Anàlisi de Coneixement de Waikato) i és un paquet desenvolupat des de la Universitat de Waikato a Nova Zelanda en 1993 i ha estat modificat i mantingut des de llavors fins a l'actualitat per fer front a problemes d'aprenentatge automàtic i mineria de dades en general. WEKA és programari lliure i està adscrit a la llicència GNU General Public License. Està escrit en *Java*, la qual cosa suposa un dels seus principals inconvenients per al seu ús en aplicacions enfocades a les altes prestacions però resulta molt útil per a propòsits pedagògics per la seua diversitat de mètodes de mineria de dades integrats i com a referència per a comprovar la bonança de resultats en noves implementacions.

WEKA es pot emprar tant a través de línia de comandaments com mitjançant una interfície gràfica desenvolupada a tal efecte. A la Figura 4 es pot vore un exemple de l'entorn gràfic de WEKA.

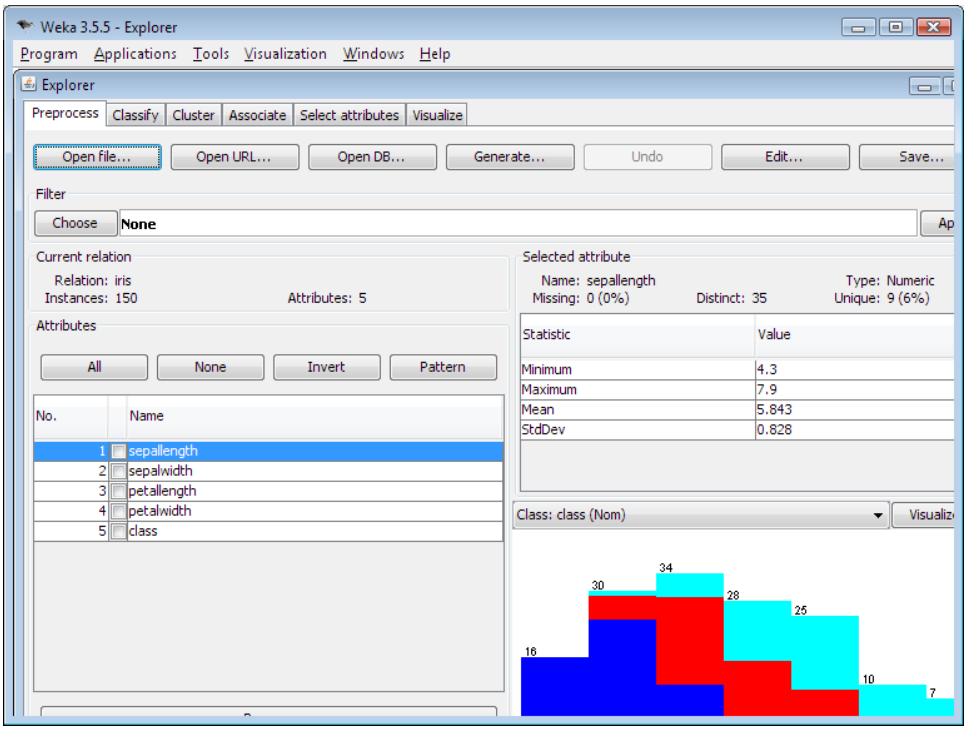

Figura 4: Vista d'una sessió de WEKA 3.5.5

WEKA conté una completa col·lecció de ferramentes de visualització i algorismes per a l'anàlisi de dades i modelat predictiu, juntament amb interfícies gràfiques d'usuari per a un accés senzill a aquestes funcionalitats. Originalment estava escrit en *C* però prompte es va migrar a *Java* (en 1997).

Els principals punts forts de WEKA es podrien resumir en els següents ítems:

- Accés lliure sota la llicència GNU General Public License.
- Fàcilment portable gràcies a que està totalment implementada en *Java* i pràcticament es pot executar en qualsevol plataforma computacional.

- Conté una ampla col·lecció de processament de dades i tècniques de modelat.
- És senzill d'usar per als no experts gràcies a la interfície gràfica inclosa.

2.3.3. BLAST

La primera de versió a analitzar del conjunt de programes de tipus BLAST [34] serà el programa homònim (http://www.ncbi.nlm.nih.gov/BLAST/), desenvolupat pel NCBI [35] (*National Center for Biotechnology Information*), el qual pren una seqüència a consultar i fa una recerca d'aquesta contra una base de dades seleccionada per l'usuari. El programa s'encarrega d'alinear la seqüència d'entrada contra cada seqüència de una base de dades de referència. Els resultats són disposats en un formulari en forma de llista ordenada seguida d'una sèrie d'alineaments de seqüències individuals, a més de diverses estadístiques i marcadors.

Per fer ús de BLAST, cal primer conéixer els seus paràmetres principals. Per una banda estan els diferents tipus de programes integrats en la ferramenta. Hi ha cinc programes BLAST diferents, els quals es poden distingir pel tipus de seqüència a buscar (ADN o proteïna) i el tipus de base de dades contra la qual es fa la recerca:

- BLASTP compara una seqüència d'aminoàcid contra una base de dades de proteïnes.
- BLASTN compara una seqüència d'un nucleòtid contra una base de dades de nucleòtids.
- BLASTX compara una seqüència d'un nucleòtid contra una base de dades de proteïnes.
- TBLASTN compara una proteïna contra una base de dades de nucleòtids.
- TBLASTX compara una seqüència d'un nucleòtid contra una base de dades de nucleòtids.

A més del programa a emprar, també és necessari especificar la base de dades sobre la qual volem fer la recerca. Com vorem més avant, amb l'apartat sobre el servei NCBI, hi ha moltes bases de dades disponibles i actualitzades diàriament. Una de les més comunament usades és la base de dades nr (col·lecció de seqüències "no redundants" de GenBank i altres bancs de seqüències).

L'altre paràmetre fonamental és la seqüència d'entrada, o fitxer de seqüències d'entrada. Han d'estar en el format estàndard anomenat FASTA o número GI (número de registre assignat a la seqüència, si el té).

Hi ha altres paràmetres importants a ajustar. Entre ells, destaquen el E-value i el Filter. El E-value (*Expect value*) és el llindar d'importància estadística per a coincidències amb seqüències de la base de dades. El valor per defecte sol ser 10, encara que és un valor alt i pot comportar l'aparició d'encerts casuals. El funcionament de l'algorisme amb aquest

paràmetre és simple; si la significança estadística associada a un encert és major que el E-value, l'encert no serà reportat. Augmentant el valor, estem forçant al programa a reportar resultats menys significatius. El valor Filter emmascara segments de la seqüència d'entrada que tenen una complexitat composicional baixa (és a dir, regions de composició esbiaixada, com ara repeticions de baix període).

A banda de tots els paràmetres vistos, també es pot triar el nombre de hits que han d'aparéixer al resultat, o el format en què volem el propi resultat.

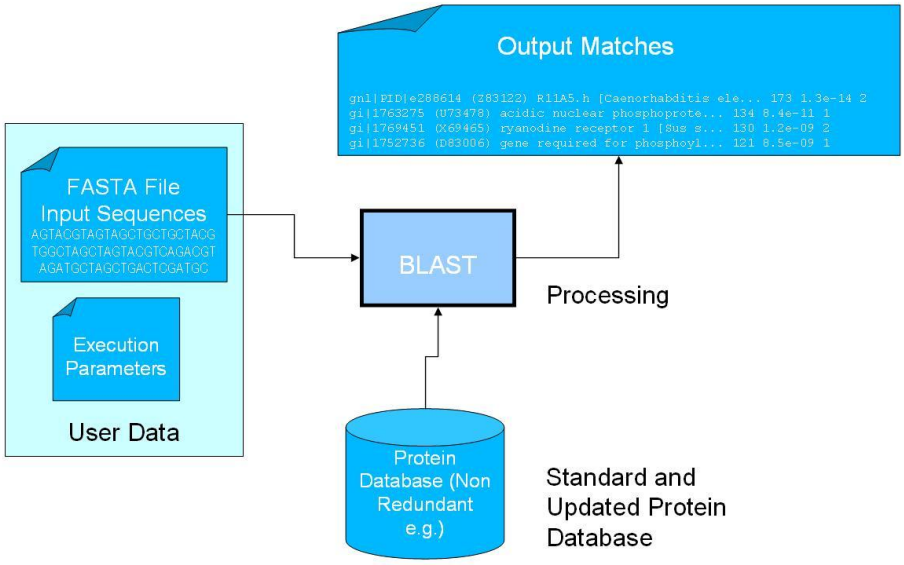

Figura 5: Procés BLAST genèric.

La implementació actualment mantinguda de BLAST és part del NCBI, i esta feta mitjançant *C++*. El nucli de l'algorisme està escrit en *C* i hi ha una API en *C++* documentada. També hi ha una API en *C* per a compatibilitats amb el *toolkit* en *C*, però ja no s'està mantenint la part en *C*. El codi font es pot descarregar via FTP o HTML.

BLAST toolkit conté diversos executables disponibles. Entre els diferents programes que podem gastar, ens centrarem en els dos més útils per als nostres propòsits:

- *blastall*: Aquest és l'executable encarregat de fer les comparacions entre unes seqüències d'entrada i una base de dades. Els principals paràmetres són el tipus de programa a gastar (*blastn*, *blastp*, *blastx*, *tblastn*, *tblastx*), la base de dades contra la que fer les recerques (nr per defecte), el fitxer de seqüències d'entrada (ruta completa al fitxer o stdin), el fitxer de resultats, el E-value (el *expect value*), el format d'eixida (XML, ASCII o altres) i el nombre de BLAST hits.
- *formatdb*: el propòsit de *formatdb* és el de formatar bases de dades d'entrada de proteïnes o nucleòtids abans de ser emprades en una recerca. La base de dades font ha d'estar o en format FASTA o en ASN.1. El fitxer font de la base de dades cal ser formatat una sola vegada. Un cop formatada la base de dades, es treballarà directament sobre aquesta. Els paràmetres bàsics són el fitxer amb

la base de dades, el nom de la base de dades formatada, el tipus de base de dades i el seu format.

2.3.4. MPIBLAST

MPIBLAST [36] és una paral·lelització del BLAST de NCBI, de distribució pública, lliure i oberta. MPIBLAST fa una segmentació de la base de dades BLAST escollida i la distribueix al llarg dels nodes de un cluster, permetent així que les recerques BLAST siguen processades en diversos nodes de forma simultània. MPIBLAST està basat en MPI i funciona sobre *Linux*, *Windows* i diverses variants de *Unix*. Per tant, MPIBLAST es pot considerar com una ferramenta molt útil que ens aprofitarà de base en el nostre projecte. A pesar de les seues virtuts, també té com a desavantatges un mal comportament amb problemes grans. Des d'aquest punt de vista, es fa imprescindible l'ús de tecnologia *Grid* per afrontar problemes grans amb garanties i també construir una interfície que interaccione amb BLAST2GO [37] [38]. Com a grans avantatges de MPIBLAST respecte al tradicional BLAST podem comentar dues d'importants. En primer lloc el fet que MPIBLAST divideix la base de dades al llarg dels diversos nodes en un *cluster*. Això permet que els segments de la base de dades poden ser així més menuts i així residir en la memòria cau, dotant d'una acceleració significativa deguda a l'eliminació de l'accés a disc. Per altra banda, permet als usuaris BLAST traure avantatge de l'eficiència fins i tot en *clusters* heterogenis, ja que les demandes de comunicació entre processadors són molt baixes.

A més de les acceleracions super-lineals [39] obtingudes en recerques contra bases de dades molt grans, també hi ha una gran millora amb el temps de latència per recerca. Els requeriments de MPIBLAST no són massa exigents. Cal tindre una implementació MPI instal·lada (funciona bé amb les dues implementacions lliures més importants, com són MPICH i LAM/MPI). A les configuracions més habituals, MPIBLAST necessita també que les computadores tinguen algun directori d'emmagatzemament compartit. Aquest pot ser tant una partició muntada NFS, una compartició SAMBA, AFS, o qualsevol altre tipus de sistema de fitxers compartit per xarxa. La localització dels directoris compartits després s'especifica en un fitxer de configuració allotjat al directori home. Per tal de construir MPIBLAST des de les pròpies fonts, és també necessari compilar una versió adequada del *Toolbox* de NCBI, disponible via *internet*.

En la versió actual de MPIBLAST, s'està en el número 1.4.x. Els executables són molt semblants als de la versió seqüencial. Per als nostres propòsits ens centrarem en dos d'ells:

- *mpiblast*: és l'equivalent a *blastall*. Els paràmetres són pràcticament els mateixos, pel que no és necessari fer grans canvis en la línia de comandaments.
- *mpiformatdb*: és l'equivalent a *formatdb*. També comparteix els mateixos paràmetres amb el respectiu executable seqüencial.

En la secció de resultats d'aquest treball, s'ha fet una anàlisi sobre l'escalabilitat de *mpiblast* i la conveniència del seu ús comparada amb l'aplicació seqüencial *blastall*.

2.3.5. MrBayes

MrBayes [40] és un programa per a la inferència bayessiana de filogènia [41]. El programa té una interfície de línia de comandaments i és capaç d'executar-se en una gran varietat de plataformes, entre d'altres en sistemes basats en *Linux, Windows* i *Mac*. El que resultaria raonable és que la computadora fora suficientment potent, tant en còmput com en memòria RAM (ja que depenent de la grandària de la matriu de dades, el programa podria requerir centenars de megabytes de memòria). *MrBayes* està optimitzat per ser el més ràpid possible però no per minimitzar els requeriments de memòria.

La construcció d'arbres d'evolucions és ja una part estàndard l'anàlisi de seqüències exploratori. Els mètodes bayessians per estimar arbres han estat recentment proposats com a un mètode més ràpid que incorpora la potència dels models estadístics complexos en el procés.

Però els mètodes bayessians no són els únics que es poden gastar sinó que n'hi ha molts i podríem enunciar els més importants i valorar breument els avantatges i desavantatges de les distintes aproximacions. Els més importants serien:

Mètode	Avantatges	Desavantatges	Programari
Unió de veïns	Ràpid	Es perd informació en la compressió de seqüències en forma de distàncies. En seqüències divergents, pot ser molt costós d'obtenir estimacions fiables de les distàncies de parells.	PAUP, MEGA, PHYLIP.
Parsimònia	Suficientment ràpid per a l'anàlisi de centenars de seqüències. Robust si les branques són curtes.	Perd prestacions si hi ha una variació substancial en les longituds de les branques.	PAUP, NONA, MEGA, PHYLIP.
Evolució mínima	Gasta models per a corregir canvis no vistos.	Les correccions de la distància poden fallar quan les distàncies dels canvis evolutius són grans.	PAUP, MEGA, PHYLIP.
Màxima versemblança	La versemblança captura el que les dades diuen sobre la filogènia sota un model donat.	Pot ser excessivament lent (depenent de la minuciositat de la recerca i l'accés a recursos computacionals).	PAUP, PAML, PHYLIP.
Bayessià	Té una connexió molt forta amb el mètode de màxima versemblança. Pot ser una forma més ràpida d'avaluar el suport als arbres que no el mètode de màxima versemblança.	Les distribucions inicials per als paràmetres han de ser especificades de forma obligatòria. Pot ser difícil de determinar si la cadena de Markov MonteCarlo (MCMC) ja ha estat suficient temps executant-se.	MrBayes, BAMBE.

Taula 1: Resum de les característiques dels principals mètodes filogenètics.

Un cop ja estan situades les diferents alternatives en quant a mètodes filogenètics, seria interessant veure algunes de les seues principals aplicacions. Algunes de les més importants podrien ser:

- Detecció d'ortologia i paralogia.

  La filogenètica es gasta de forma habitual per ordenar la història de duplicacions de gens per a famílies de gens. Aquesta aplicació està actualment inclosa fins i tot en exàmens preliminars de dades de seqüències; per exemple, l'anàlisi inicial del genoma del ratolí incloïa arbres d'unió de veïns per a identificar duplicacions en citocrom P450 i altres famílies de gens.

- Estimació de la divergència filogenètica.

  Les implementacions bayessianes de nous models van permetre a Aris-Brosou i Yang d'estimar quan un animal divergia filogenèticament sense assumir un rellotge molecular.

- Reconstrucció de proteïnes antigues.

  Chang et al. gastaven la màxima versemblança (ML) per a reconstruir les seqüències de pigments visuals en els darrers ancestres comuns dels pardals i els caimans; la proteïna va ser sintetitzada al laboratori de forma exitosa.

- Recerca dels residus que són importants a la selecció natural.

  Localitzacions d'amino-àcids en la superfície del virus de la grip que són objectius del sistema immunològic poden ser detectats per un excés de substitucions sense sinònim. Aquesta informació pot ajudar a la preparació de les vacunes.

- Detecció de punts de recombinació.

  Els nous mètodes bayessians poden ajudar a determinar quines variants o subtipus del virus de la immunodeficiència humana 1 (HIV-1) sorgeixen de la recombinació.

- Identificació de mutacions amb possibilitats d'estar associades amb una malaltia.

  La mancança de dades estructurals, bioquímiques i funcionals de molts gens implicats en malalties significa que no està clar quines mutacions malsentit són importants. Fleming et al. gastaven filogenètica bayessiana per tal d'identificar mutacions malsentit en regions conservades i regions sota selecció positiva en el gen BRCA1 del càncer de pit. Aquestes dades els permetien prioritzar entre les mutacions per a futurs estudis funcionals i poblacionals.

- Determinació de la identitat de nous patògens.

  L'anàlisi filogenètic ara mateix es realitza de forma rutinària després l'ampliació de la tècnica coneguda com la reacció en cadena de la polimerasa (PCR) de fragments genòmics de patògens prèviament desconeguts. Aquestos anàlisis fan possible una ràpida identificació de tant el Hantavirus com del virus del Nil occidental.

2.3.6. CLUSTAL

Clustal [42] és un programa amplament gastat per a la alineament de múltiples seqüències per a nucleòtids o proteïnes. Aquest programa produeix alineament de seqüències múltiples amb significat biològic a partir de seqüències divergents. *Clustal* calcula el millor candidat per a les seqüències seleccionades, i les alinea de forma que les identitats, similituds i diferències són evidenciades. Les relacions evolucionàries també poden ser vistes mitjançant els diagrames cladístics i filogenètics.

Existeixen dues versions principals, que són *ClustalW* [43](una interfície per línia de comandaments) i *ClustalX* [44](amb interfície gràfica per a *Windows*, *Mac OS* i *Unix/Linux*). Les dues versions es poden obtenir al servidor ftp de l'Institut Europeu de Bioinformàtica (ftp://ftp.ebi.ac.uk/pub/programari).

Aquest programa accepta una ampla varietat de formats d'entrada, entre els que destaquen NBRF/PIR, FASTA, EMBL/Swissprot, Clustal, GCC/MSF, GCG9 RSF i GDE. El format d'eixida pot ser un o diversos dels següents: Clustal, NBRF/PIR, GCG/MSF, PHYLIP, GDE, NEXUS.

En l'alineament múltiple de seqüències hi ha tres passos fonamentals:

- Fer l'alineament entre parells.
- Crear un arbre filogenètic (o gastar-ne un proporcionat per l'usuari).
- Gastar l'arbre filogenètic per fer l'alineament múltiple.

Aquestos passos són els realitzats quan se tria "Fer alineament complet", però hi ha altres opcions com ara fer alineament a partir d'un arbre de guia o produir només l'arbre de guia.

Els alineaments entre parells es calculen per a totes les seqüències contra totes les seqüències, i les similituds s'enregistren dins d'una matriu. Aquesta és aleshores convertida en una matriu de distàncies, on la distància mesura la distància evolutiva entre cada parell de seqüències

Des d'aquesta matriu de distàncies, es construeix un arbre de guia o filogenètic, per a la qual cosa dos parells de seqüències han de ser alineades i combinades amb alineaments previs. La construcció es fa mitjançant un algorisme de *clustering* d'unió de veïns. Les seqüències són progressivament alineades a cada punt de la branca, començant des del parell de seqüències més proper.

Per a la majoria d'usuaris, l'alineament de seqüències no requerirà més configuració que la que hi ha per defecte, però ocasionalment pot ser interessant adaptar la configuració als requeriments propis del problema. Els paràmetres més interessants són el *gap opening penalty* i el *gap extension penalty*.

Existeix una versió distribuïda i paral·lela anomenada ClustalW-MPI [45] que és la corresponent implementació en MPI de *ClustalW*. Els passos bàsics que s'han comentat anteriorment (alineament de parells, generació de l'arbre de guia i alineament progressiu) són paral·lelitzats per tal de reduir el temps d'execució. El programari gasta la llibreria de pas de

missatges anomenada MPI (*Message Passing Interface*) i s'executa en *clusters* de treball distribuïts de la mateixa manera que en computadores paral·leles tradicionals. Està escrit en *ISO C* i és codi obert.

2.3.7. Aplicacions bioinformàtiques i tecnologia Grid

Als punts anteriors es parla sobre l'estat actual de les tecnologies i ferramentes que tenen una relació directa amb el treball present. En aquest punt la idea és fer en un breu anàlisi, a través de tres de les aplicacions bioinformàtiques abans presentades, veient les seues possibilitats i idoneïtat per al desplegament en una infraestructura *Grid*. Tant el mètode dels K veïns més pròxims com la seua versió en WEKA no seran analitzats ja que el cas estaria representat per l'anàlisi feta a l'aplicació BLAST i per evitar redundàncies i clarificar l'objecte de l'estudi, s'ha limitat aquest a tres aplicacions de les *heretades*.

Aleshores, l'anàlisi estarà relacionada amb el que es podria anomenar, creant un neologisme tecnològic, *gridificabilitat*. Així, es podria dir que una aplicació és *altament gridificable* si s'adapta a la perfecció a un entorn *Grid* o *baixament gridificable* en cas que la seua adaptació siga impossible o molt complicada.

Les aplicacions a comentar són tres: BLAST, CLUSTAL-W i Mr.Bayes. Les tres estan relacionades amb l'alineament de seqüències de proteïnes i nucleòtids, però la primera l'alineament és senzill i en les altres dues és múltiple. O siga, el BLAST fa una recerca d'una seqüència contra un grup (normalment gran) de seqüències a mode de base de dades, mentre que CLUSTAL-W i Mr.Bayes el que fan és comparar un grup de seqüències entre elles mateixa, totes alhora.

Pel que fa al grau de *gridificabilitat* (o siga, la mesura en què s'adapta a un entorn *Grid*), BLAST ofereix dues versions, la versió seqüencial *blastall* i la versió MPI anomenada *mpiblast*. El procés BLAST té per entrada un conjunt més o menys gran de seqüències que s'hauran de comparar amb unes seqüències de referència en forma de base de dades. El particionament de les dades es pot fer en dos sentits: particionant les seqüències d'entrada o particionant les seqüències de la base de dades. La primera aproximació té la senzillesa que no cal fer un post-processament de les dades d'eixida, ja que l'eixida serà fruit d'un processament complet, i la segona aproximació sí que necessita d'un post-processament, ja que cal reordenar les eixides i descartar-ne en cas que siga necessari. La *gridificabilitat* de BLAST és molt alta, ja que hi ha dos tipus de particionament i la dependència entre els sub-treballs generats és o nul·la o baixa. En cas d'un ús de mpiBLAST, no hi ha cap tipus de limitació en el nombre de nodes (que no siga el propi nombre de seqüències de la base de dades de referència).

En el cas de CLUSTAL-W, tenim processos que tenen per entrada un conjunt de seqüències que no es poden particionar de manera que es creen treballs independents, com al cas anterior. En aquest cas, també hi ha una versió paral·lela en MPI que pot contribuir a millorar el comportament temporal però el nombre de processos MPI té una limitació actual, que es

pot expressar a l'Equació 2**¡Error! No se encuentra el origen de la referencia.**, on N és el nombre de seqüències del fitxer d'entrada.

$$p_{MPI} < \frac{N \cdot (N-1)}{2}$$

Equació 2: Relació entre el nombre de processos MPI i el nombre de seqüències d'entrada

Amb tot, CLUSTAL-W té, en principi, un grau de *gridificabilitat* baix, però un grau de paral·lelització bo, ja que es pot considerar que no hi ha una limitació en el nombre de processos MPI de cada treball. Afortunadament, els processos CLUSTAL-W no sempre vindran assoles sinó que sovint seran producte d'un pre-procés previ d'un conjunt de seqüències que, després d'un alineament BLAST, s'aprofita l'eixida per fer un alineament múltiple amb CLUSTAL-W. Aquest és el cas d'una anàlisi metagenòmica, on es fa l'alineament BLAST, se filtren les eixides creant un subconjunt menut, es fa un alineament múltiple amb CLUSTAL-W i posteriorment es crea un arbre filogenètic, per posar un exemple.

Mr.Bayes, com ja s'ha explicat a l'apartat de l'estat de l'art, vindria a cobrir el mateix espai que CLUSTAL-W però amb un mètode diferent, la inferència bayessiana, i amb unes característiques també diferents, com una millor ressolució dels resultats però també un cost computacional molt superior al de CLUSTAL-W.

Per altra banda, Mr.Bayes en la seua versió MPI té unes limitacions molt superiors a les de CLUSTAL-W, on ara ja no es depén de les seqüències d'entrada sinó del nombre d'execucions i del nombre de cadenes (paràmetres propis de l'execució). Així, ara el nombre de procés MPI prén la forma de l'Equació 3**¡Error! No se encuentra el origen de la referencia.**.

$$p_{MPI} \leq nruns \cdot nchains$$

Equació 3: Relació entre el nombre de processos MPI i alguns paràmetres de l'execució de Mr.Bayes.

Per tant, l'escalabilitat de Mr.Bayes és notablement inferior a la de CLUSTAL-W. Això es pot sumar als arguments expressats ja amb el cas de CLUSTAL-W sobre la seua baixa *gridificabilitat*. Tot junt, desaconsella molt l'ús d'aquesta ferramenta en un entorn *Grid* i convé buscar-li alternativa en cas que siga possible.

Fent un ràpid repàs es pot concloure que és important que el tipus de treballs relacionats amb les aplicacions siguen fàcilment divisibles en sub-treballs, com al cas de BLAST, i que les aplicacions siguen escalables pel que fa al nombre de nodes en una paral·lelització MPI, com és el cas de CLUSTAL-W. El pitjor cas sempre serà aquell en què no és possible dividir el treball en subtreballs i hi ha un límit en el nombre de processos MPI utilitzables en una execució paral·lela.

2.4. Casos d'estudi

En els experiments que es realitzaran en el marc d'aquesta memòria s'empraran una sèrie de casos d'estudi per dotar de realisme als propis experiments i també per a ser d'utilitat als projectes en què es troba

immers el treball desenvolupat. Als experiments realitzats s'han gastat metagenomes reals, com el del mar dels sargassos o tretze metagenomes de microbiota intestinal humana per als experiments basats en BLAST, tant les versions seqüencials com les paral·leles. Per a la part d'integració dels tres paradigmes de computació paral·lela s'ha emprat una base de dades sintètica per a aplicar el mètode dels K veïns més pròxims i aprofundir en les seues característiques no es considera necessari.

2.4.1. Metagenoma del mar dels sargassos

El mar dels Sargassos és una regió de l'oceà atlàntic septentrional situada entre els meridians 70º i 40º O i els paral·lels 25º i 35º N, amb una superfície total aproximada de 3.500.000 quilòmetres quadrats. Es caracteritza per una absència freqüent de vent, corrents marines i abundància de plàncton i algues (que formen autèntics boscos marins superficials que s'estenen d'horitzó a horitzó).

A les aigües superficials, on arriba la llum, abunda el plàncton vegetal, que consumeix sals com els fosfats i nitrats. Degut a la diferència de densitat, l'aigua de la superfície gairebé no es barreja amb l'aigua freda i rica en minerals de les capes inferiors, que podrien restaurar les sals consumides. Per aquesta raó, a les regions superiors del mar dels Sargassos pràcticament no hi ha vida animal, i no tindria cap interés biològic si no fora per l'alga que li dóna el nom, el sargàs (gènere sargassum), que forma grans camps, plens d'organismes marins. Aquesta riquesa biològica és una font important per a la investigació farmacèutica.

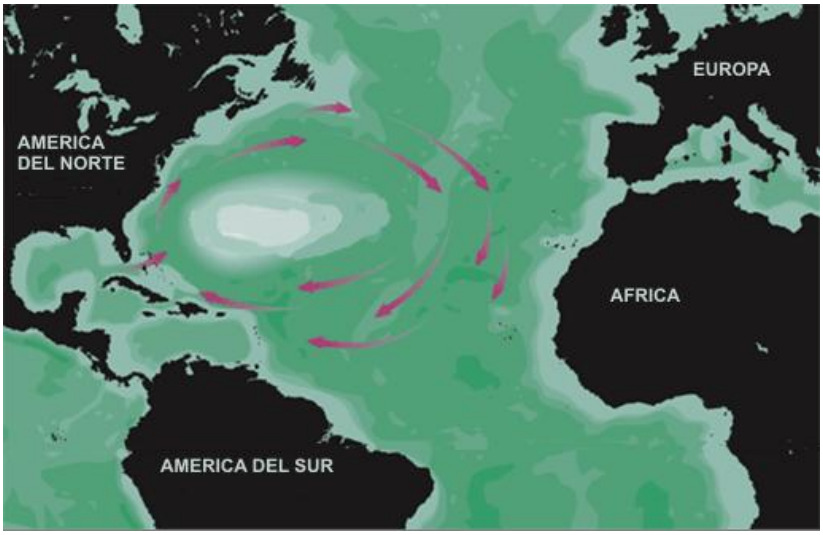

Figura 6: Ubicació geogràfica del mar dels sargassos.

El metagenoma del mar dels sargassos emprat està codificat a un fitxer amb 811.372 seqüències i que ocupa un total de 242,3 Mbytes comprimit amb *gzip*.

2.4.2. Metagenomes de microbiota intestinal en humans

Per microbiota intestinal es coneix a la microflora i microfauna en conjunt, concretament a la gran varietat de bacteris que conviuen als intestins d'humans i altres animals. Aquest conjunt de bacteris ajuda al seu amfitrió a metabolitzar molts nutrients que serien indigeribles d'altra manera. També estan relacionats amb altres funcions importants, com ara la maduració i modulació de la resposta immunològica del seu amfitrió, i la prevenció d'infeccions per part de patògens bacterians. Degut a l'estreta relació entre el conjunt de bacteris i algunes de les funcionalitats bàsiques el cos humà, hi ha un interés creixent per conéixer millor tant la identitat dels bacteris com la seua funció dins dels processos vitals associats.

Per tot això, l'anàlisi metagenòmica pot contribuir a aclarir molts dubtes relacionats amb aquest tipus de bacteris com, per exemple, la identificació de quina és la relació entre bacteris i processos vitals o fer una recerca de les diferències entre diversos metagenomes de persones de sexes, edats i saluts diferents.

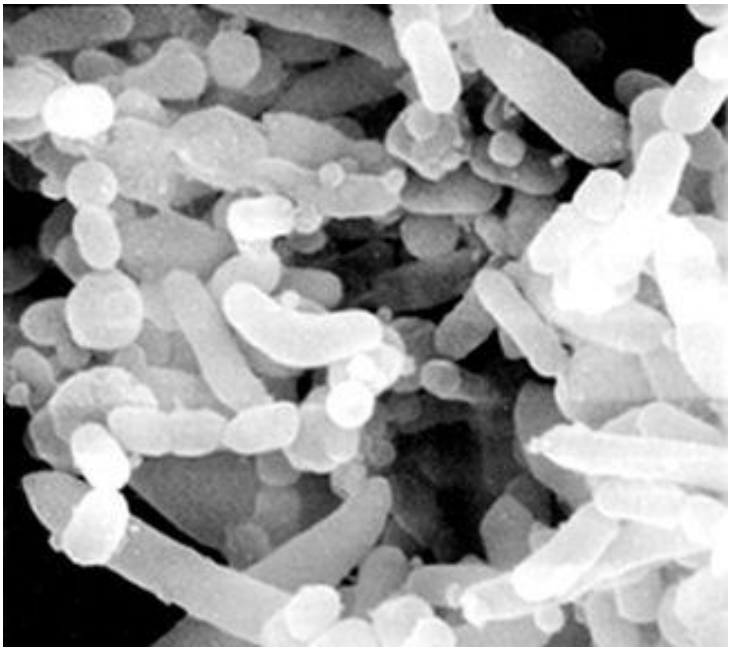

Figura 7: Imatge microscòpica del bacteri intestinal Bacteroides Thetaiotaomicron.

El cas d'estudi es presenta en forma de tretze metagenomes corresponents a tretze humans de diferents característiques de salut, edat i sexe, degudament anonimitzats. En suma, es tracta d'un total de 473 Mbytes i 353.801 seqüències.

3. INTEGRACIÓ MPI DINS D'UNA INFRAESTRUCTURA GRID

En aquest apartat es farà una breu revisió de les possibilitats que actualment es troben quan s'intenta combinar MPI, com a llibreria estàndard per a la programació paral·lela en memòria distribuïda, i la tecnologia *Grid*. Per això primerament es parlarà de la situació concreta dintre de la infraestructura *Grid* d'EGEE amb l'ús del *middleware* gLite 3 i LCG-2. A continuació s'analitzaran dues aproximacions *Grid* de MPI que han tingut una repercusió relativa com són MPICH-G2 i PACX-MPI, amb les seues característiques i diferències principals, i es farà un breu esment a un dels projectes europeus que estan tractant el tema específicament.

3.1. L'integració dintre de EGEE

Les aplicacions computacionalment intensives requereixen de tècniques basades en la computació d'altes prestacions (HPC) per a la seua execució eficient. La computació en *Grid* (com pot ser la infraestructura de EGEE) complementa al HPC permetent l'accés a nous recursos de computació i a noves formes de computació (sistemes híbrids que combinen alta productivitat i altes prestacions). Després de conéixer les principals ferramentes de computació paral·lela a apartats anteriors, convé conéixer també els requisits de les aplicacions paral·leles per a analitzar quina serà la funcionalitat necessària per als principals *middlewares Grid*. Òbviament, ens centrarem en MPI com a principal ferramenta de computació paral·lela en entorns de memòria distribuïda i en Globus Toolkit 4.x i *gLite* 3.0 (*LCG-2*) com *middlewares Grid*.

L'ús de MPI en el *Grid* no és trivial. La guia d'usuari de *gLite* 3.0 [46] es troba un apartat anomenat *MPI Jobs*, on diu textualment:

"La interfície de pas de missatges (MPI) és una llibreria estàndard comunment emprada per a la programació paral·lela. El WMS de gLite dóna suport de forma nativa al llançament de treballs MPI, els quals són treballs composats per un nombre de processos executant-se en diferents WNs al mateix CE. Tanmateix, aquest suport és encara experimental i la seua funcionalitat no es descriurà en aquesta guia d'usuari. La font més completa d'informació sobre treballs MPI en Grid es troba a la MPI Wiki page [47]."

A la guia d'usuari de *LCG-2* també hi ha un apartat anomenat *MPI Jobs* però en aquest cas hi ha una descripció un poc més concreta del que ha de ser un treball MPI sobre *LCG-2*. Per una banda es parla de l'estructura del fitxer JDL i també es parla del *shell-script* on es llança el programa MPI.

Tant és l'interés que ha suscitat el tema, que s'han creat diversos grups de treball entre els que destaca el EGEE-TCG (grup de coordinació tècnica d'EGEE) sobre la integració de MPI dins de la infraestructura *Grid* del projecte EGEE-II [48]. Aquest grup, liderat pel professor Stephen Childs

de la Universistat de Dublín | Trinity College, es va crear en desembre de 2005 i la seua tasca va concloure en desembre de 2006. Del treball d'aquest grup han nascut alguns documents tècnics amb els punts obligatoris, recomanables i desitjables per a que la integració de MPI en la infraestructura EGEE-II fora plena [49]. De fet, aquest grup és el creador de la pàgina esmentada a la guia d'usuari de *gLite* 3.0.

Pel que fa als requisits per ala computació *Grid* des del punt de vista de l'usuari, hi ha una sèrie de punts definits pel grup de coordinació tècnica de EGEE (EGEE-TCG) format en Q1/2006 per a l'execució de treballs paral·lels en Grid:

- Obligatoris:
  - Els usuaris han de preparar el treball abans del llançament de l'executable MPI (compilació i/o preparació d'entrades).
  - Alguns treballs necessiten un directori compartit. Durant l'execució del treball, alguns fitxers del directori poden ser actualitzats. Aquests canvis han de ser visibles a tots els processos.
  - S'ha de poder donar suport a diverses versions MPI. Açò inclou treballs que precisen de diverses versions (MPI-1, MPI-2) i diferents implementacions (MPICH-1, MPICH-2, LAM, etc.).
- Recomanables:
  - Algunes aplicacions necessiten la utilització de MPI a través de múltiples màquines.
  - Algunes aplicacions necessiten de compiladors comercials
  - Algunes aplicacions desitjarien un control detallat de la distribució de treballs, especificant el nombre de nodes i el nombre de CPUs per node.
- Suggeriments:
  - Algunes aplicacions necessiten la invocació de serveis *Grid* des dels processos fills. S'ha de gestionar l'accés al *proxy*.

També hi ha referències al mateix document sobre el punt de vista de l'administrador:

- Obligatoris:
  - Han de comprendre's els requeriments d'un directori d'usuaris compartit. Falta per especificar si ha de ser $HOME el directori compartit o pot ser qualsevol altre o quins fitxers haurien de replicar-se als nodes en cas que no hi haja un sistema d'arxius compartit.
- Recomanables:
  - Cal assegurar-se que els treballs MPI són compatibles amb tots els gestors de treballs (lcgpbs, pbs, etc.).
  - La majoria dels recursos que donen suport a MPI, suportaran una barreja de treballs paral·lels i seqüencials. Ha de realitzar-se una planificació eficient en aquest cas. El LRMS ha de tenir accés a la descripció del treball.

  - S'han d'utilitzar tècniques de metaplanificació eficients per a treballs MPI, que consideren el nombre de processadors disponibles, el nombre total de processadors, etc.
  - La informació proporcionada pel LRMS ha d'integrar-se al sistema d'informació de la màquina i proporcionar la informació correcta.
  - Si es dóna suport a MPI a través de diferents nodes, aleshores han de determinar-se els requeriments IP.
- Suggeriments:
  - Hauria d'evitar-se que el RB assumisca informació sobre el recurs remot. Algunes màquines prefereixen utilitzar *Torque* en lloc de *PBS*, així com *mpiexec* en lloc de *mpirun*.
  - Caldria donar suport a una àmplia varietat de sistemes de cues, com ara *SGE*, *Torque* i inclús *LSF*.
  - S'hauria de disposar d'un entorn d'usuari estàndard. Per exemple, disposar de variables d'entorn que indicaren la localització d'un disc compartit. A més, caldria que hi haguera variables d'entorn per indicar la ruta a aquestes implementacions.

Fruit dels treballs del mateix grup de coordinació tècnica, també hi ha molts documents on es redunda en les principals demandes i carències dels actuals *middlewares*. En els informes, s'analitzen els components de *gLite* (WMS, CE,WN i UI) i de ferramentes de configuració per comprovar si les ferramentes de MPI estan instal·lades i correctament publicades.

A més d'aquestes qüestions relacionades amb aspectes tècnics, més endavant, a la secció del desplegament d'experiments en EGEE-II, es podrà veure una gràfica que dóna fe de la baixa quantitat de recursos de còmput de la infraestructura *Grid* de EGEE-II que donen suport a treballs paral·lels que fan ús de MPI. De fet, per a treballs amb un requeriment de nodes menuts (menys de 10 nodes), el nombre màxim de recursos disponibles sol ser la quarta part dels disponibles per a treballs seqüencials.

3.2. Projectes i implementacions MPI per a Grid

Les aplicacions d'una versió MPI per a *Grid* estan relacionades amb la capacitat de la pròpia infraestructura *Grid*, podent dotar als programes MPI de més capacitat computacional, més memòria i més ample de banda de la xarxa. Lògicament, no tot van a ser avantatges i s'haurà de lluitar contra factors com l'heterogeneïtat, la barrera de les diferents organitzacions administratives que componen un *Grid*, els possibles tallafocs que es troben al camí, les altes latències degudes a la distància física i lògica i la tolerància a errades. En aquest context hi ha dues implementacions com són MPICH-G2 i PACX-MPI que intenten afrontar el repte.

3.2.1. MPICH-G2

MPICH-G2 és una implementació de l'estàndard MPI-1.1 per a l'execució d'aplicacions paral·leles basades en MPI a través de distintes màquines. Aquestes màquines poden ser de diferents arquitectures i la conversió dels tipus de dades es fa de manera transparent a l'usuari. La comunicació es fa a través de TCP entre les màquines de diferents *clusters* i el *middleware* MPI instal·lat a cada *cluster* per a comunicacions dins del propi *cluster*.

La llibreria MPICH-G2 és la segona versió del projecte MPICH-G, que va ser millorat incrementant l'ample de banda, reduint la latència per a missatges intra-màquina, fent un ús més eficient dels *sockets*, es donava suport al tipus de dades MPI_LONG_LONG i a les operacions de fitxers MPI-2 i també suport a aplicacions escrites en *C++*.

La base de MPICH-G2 és la llibreria MPICH, amb l'afegit que s'usen els mecanismes de *Globus Toolkit* per a diverses tasques com les d'autenticació, autorització, entrada/eixida, creació de processos i monitorització. Està preparada per treballar amb els serveis Pre-WS de *Globus Toolkit*, cosa que la fa compatible amb les noves versions.

Per al programador el procés d'implementació de programes és totalment anàloga a com es faria en un simple computador paral·lel i la seua funcionalitat està disponible com un dispositiu MPICH (*globus2*). Això fa que MPICH-G2 siga capaç de fer front a problemes com l'autenticació entre màquines, interactuar amb diversos sistemes de cues, crear processos de forma coordinada, possibilitar la comunicació d'estructures heterogenies, transmetre l'executable i recollir l'eixida estàndard.

A priori és evident la millora que suposa aquesta llibreria, ja que no ha de canviar pràcticament res en la forma d'implementar programes en MPI i fàcilment es té accés a tot un seguit de recursos més enllà del domini administratiu propi. Això també comportarà problemes, com és l'impacte en les prestacions degut a la seua alta latència i a la reducció de l'ample de banda per a les aplicacions paral·leles de gra fi on les comunicacions són un factor clau.

A més, hi ha altres problemes propis de l'estratègia triada, com pot ser el fet que MPICH-G2 s'enfoca com a sistema basat en pas de missatges on tots els nodes que conformen el sistema de computació haurien de poder comunicar-se directament i la realitat és que no sempre els nodes interns d'un *cluster* van a tindre una IP pública o simplement connexió directa a *internet*. Per últim, la utilització de *Globus Toolkit* també suposarà els problemes amb els tallafocs que ja existien en el citat *middleware Grid*, sobretot a l'hora de crear, monitoritzar i cancel·lar treballs.

3.2.2. PACX-MPI

La llibreria PACX-MPI (PArallel Computer eXtension to MPI) també permet executar aplicacions paral·leles MPI en un *Grid* computacional. Les aplicacions MPI desenvolupades han de ser recompilades amb aquesta llibreria, que implementa MPI-1.2.

Novament, hi ha dos tipus fonamentals de comunicacions, com ja passava amb MPICH-G2, les comunicacions intra-*cluster* i les comunicacions inter-*cluster*. En les intra-*cluster* es gastarà directament la distribució MPI disponible en el propi *cluster*, mentre que per a les comunicacions inter-*cluster* s'optarà per l'utilització de dos dimonis que controlen la comunicació entre els sistemes. Aquests dimonis s'implementen com a processos MPI locals, de forma que s'agrupen les comunicacions, s'evita tenir milers de connexions obertes entre processos, permet controlar la seguretat d'una manera centralitzada i amb l'utilització de múltiples fils d'execució es pot ajudar a ampliar l'ample de banda de les comunicacions.

L'utilització de dimonis com a passarel·les en les comunicacions fa que PACX-MPI puga ser emprat en *clusters* amb nodes interns amb IP privada, contràriament al que passava amb MPICH-G2.

3.2.3. Altres projectes MPI en Grid

Pel que fa a altres activitats i projectes en computació paral·lela inter-*cluster* destaca el projecte int.eu.grid [50]. Es tracta d'un projecte europeu finançat pel VI Programa Marc [51] i coordinat pel Institut de Física de Cantàbria (IFCA), que pretén continuar l'experiència guanyada amb el projecte *CrossGrid* [52]. El seu objectiu bàsic és la posada en marxa d'una infraestructura *Grid* avançada en l'àrea de la recerca europea, especialment orientada al suport de l'execució d'aplicacions que necessiten de certa interactivitat, que actualment interopera amb la infraestructura EGEE. En aquest projecte col·laboren actualment 13 institucions de 7 països europeus. El projecte int.eu.grid utilitza PACX-MPI com a suport per a l'execució de treballs paral·lels inter-*clusters*.

4. DESPLEGAMENT D'EXPERIMENTS MASSIUS EN e-INFRAESTRUCTURES EN PRODUCCIÓ

Pel que fa referència al desplegament dels experiments massius en infraestructures científiques en producció, com EGEE o EELA, s'han hagut de seguir una sèrie de passos bàsics què, depenent del tipus d'experiment tindran algunes variacions. Resulta obvi que no serà el mateix un experiment seqüencial que un de paral·lel, o un experiment amb el programari necessari prèviament instal·lat o amb un sistema auto-contingut. També cal tindre en compte si els fitxers necessaris per a l'execució dels treballs es troben emmagatzemats en recursos *Grid* o si simplement es passen directament a la safata d'entrada de l'especificació del treball *gLite*.

A la Figura 8 es pot vore un senzill esquema amb les principals tasques que conformen el desplegament d'un experiment massiu en una e-infraestructura en producció. L'ordre indicat és una possibilitat, ja que hi ha tasques que es poden realitzar en un ordre distint, com l'especificació de treballs i la selecció d'elements de còmput, però també hi ha altres tasques que s'han de fer seqüencialment, és a dir, la selecció de recursos de còmput sempre és anterior a la selecció de recursos d'emmagatzemament.

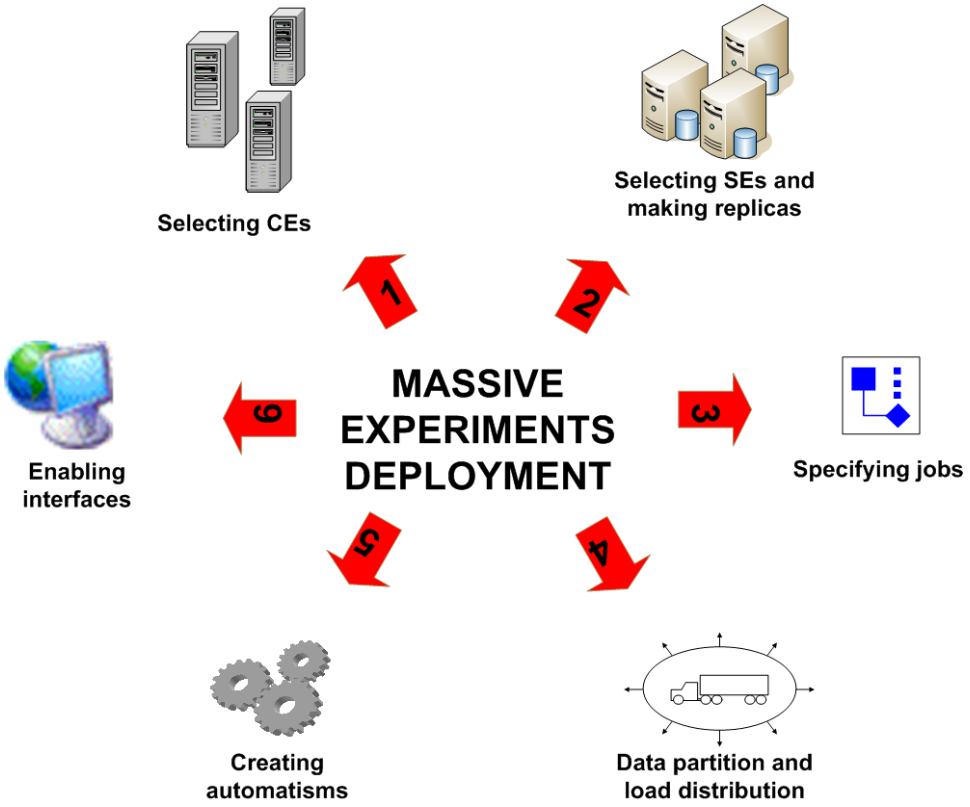

Figura 8: Esquema amb les tasques que conformen el desplegament d'un experiment massiu.

En un primer lloc, cal observar quins són els CEs que hi ha disponibles per a l'experiment. A partir del llistat de CEs definitivament triats, caldrà

decidir també quins són els SEs escollits per a albergar les distintes rèpliques dels fitxers d'entrada (bases de dades, fitxers de seqüències, etc.).

Amb els recursos escollits, començaran les primeres proves amb el *shell-script* bàsic que gestionarà tota la marxa del treball, des de la recepció de fitxers d'entrada fins a l'enregistrament dels fitxers d'eixida. L'entrada a escena del *shell-script* bàsic farà que hi haja una certa realimentació pel que fa als recursos escollits prèviament, ja que el *shell-script* pot posar de manifest problemes concrets amb els recursos.

Un cop ja hi ha uns recursos definitius i un *shell-script* apropiat per resoldre el problema, cal treballar en l'automatització del sistema i crear un servei *Grid* en cas que siga necessari.

En l'actualitat hi ha molt de treball fet per tal d'incrementar les capacitats computacionals en infraestructures Grid com la d'EGEE. Aquest és el cas del projecte WISDOM [53] i molts altres treballs, com les darreres publicacions al BionfoGRID Symposium [54], especialment aquelles que tracten el problema de l'anàlisi genòmic [55] i la integració de diverses etapes d'estudis genètics al mateix servei [56]. Aquests són simplements uns quants exemples de com d'interessant és afegir les capacitats que ofereix el Grid en aquests camps i com de prometedores són les prestacions als experiments preliminars, que es podran vore al següent capítol.

4.1. Selecció de CEs

A l'hora de triar els recursos de còmput per executar una tasca el primer que caldria fer és analitzar els recursos disponibles per a l'organització virtual a la que pertany el certificat amb que l'usuari s'identifica. Amb *LCG-2* es faria ús del comandament *edg-job-list-match* per determinar els recursos de còmput disponibles per a executar un treball concret. D'aquesta manera, es podrien dividir els treballs en seqüencials i paral·lels. Per a treballs seqüencials, en el moment de l'experiment, s'obtenien un total de 114 recursos disponibles en l'organització virtual BIOMED de EGEE-II. Si es fa el mateix experiment per a treballs paral·lels amb diferent nombre de nodes requerits, s'obté la gràfica de la Figura 9 que relaciona els nodes requerits (a l'eix de les X) amb els recursos disponibles (a l'eix de les Y).

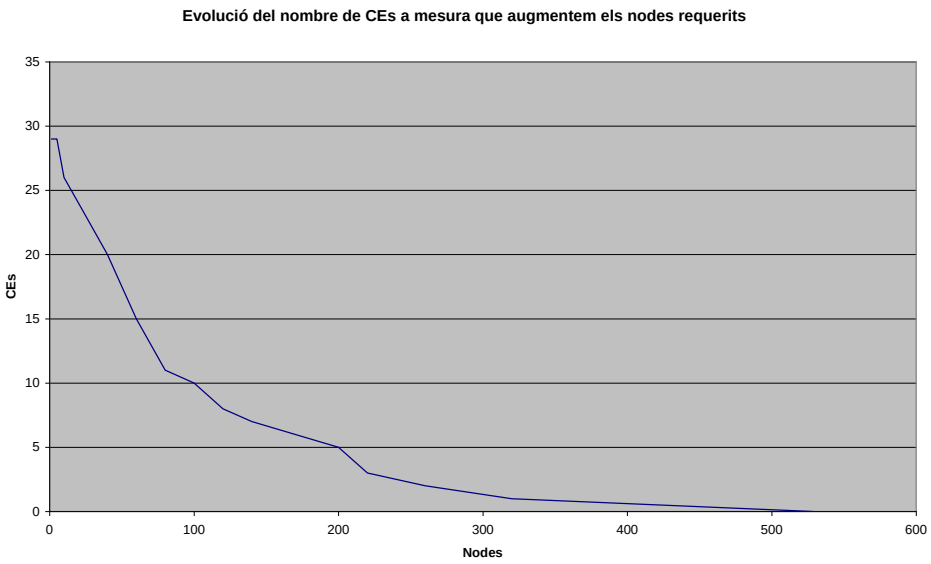

Figura 9: Evolució del nombre de CEs a mesura que s'augmenten els nodes requerits.

D'entrada s'aprecia que, només pel fet de tractar-se de treballs de tipus MPI, es redueix el nombre de recursos disponibles a 29, o siga, una quarta part dels recursos disponibles per a treballs seqüencials. La majoria dels recursos MPI donen suport a treballs amb requeriments de 40 nodes o menys. Tanmateix, n'hi ha 10 recursos amb suport per a més de 100 nodes. Açò implica la disponibilitat de fins a 1000 CPUs en total, d'entre les 17.000 CPUs disponibles de forma individual.

Un cop vist el nombre de recursos disponibles, depenent del tipus de treball, caldrà prendre una o altra decisió. Un dels primers problemes normalment faran referència a la compilació dels programes a executar. L'experiència aconsellarà pre-compilacions estàtiques, però esta tasca no sempre serà la millor, a més d'augmentar de forma considerable la grandària dels binaris. Altres vegades, degut a la simplicitat dels fonts gastats i absència (o limitació) dels enllaços a llibreries externes, fa aconsellable provar una compilació al propi recurs de còmput. Un problema comú sol l'absència de permís per a l'execució de compiladors als recursos de còmput (per exemple, *cc*, *gcc* i, sobretot, *mpicc*).

En ocasions n'hi haurà restriccions pel que fa al programari instal·lat al recurs. Açò serà un problema recurrent a qualsevol infraestructura *Grid* i, quan no siga possible la instal·lació dins del mateix treball *Grid*, caldrà posar-se en contacte amb els administradors de diferents recursos, cosa que limitarà definitivament les possibilitats a l'hora de llançar treballs *Grid* lliurement. Un exemple d'aquest darrer tipus de problemes són els treballs que fan ús de *MPIBLAST* què, com s'ha vist a l'apartat corresponent, necessita d'una certa configuració per tal d'assegurar l'optimització del programa i traure-li rendiment a aquesta versió MPI de BLAST, ja que amb una configuració dolenta, es podrien perdre prestacions i ser, en la pràctica, pitjor que la corresponent versió seqüencial.

Resulta evident que per definir un grup de recursos de còmput cal recórrer també a les proves empíriques, i així eixir de dubtes sobre els problemes en quant a configuració que tenen els diferents recursos. Aquestes proves, al mateix temps, poden aportar més dades, com ara informació sobre la

velocitat de còmput, la capacitat de memòria primària i secundària o les velocitats en transaccions d'entrada-eixida. Aquestes dades, en cas de tindre abundància de recursos, constituiran criteris objectius per triar un recurs o un altre. Altres dades interessants serà la configuració del recurs de còmput amb el seu entorn, com podria ser el seu element d'emmagatzemament més proper o els sistemes d'informació i catàleg.

Per analitzar els diferents problemes, s'ha optat per fer proves amb l'execució d'un treball *Grid* seqüencial complex amb diferents operacions dins d'un *shell-script* (en l'ordre especificat):

- descàrrega de fitxers (*wget*).
- descàrrega de fitxers des d'un SE amb el comandament (*lcg-cp* de *gLite/LCG-2)*.
- instal·lació, configuració i execució d'un programa d'alineament de seqüències genòmiques (*blastall* de NCBI).
- pujada de fitxers a un SE amb el comandament (*lcg-cr* de *gLite/LCG-2*).

Amb l'anterior treball es van llançar diversos experiments sobre tots els CEs disponibles (els prop de 114 recursos obtinguts mitjançant l'execució de *edg-job-list-match*). La taxa d'errors total obtinguda estava al voltant del 31% i es podia desglossar en errors en *wget* (21% sobre el total d'errors), en *lcg-cp* (71%), en execució del *blastall* (4%) i en *lcg-cr* (4%), significant això que un cop es trobava un error no es continuava l'execució i l'ordre era l'anteriorment exposat. Aquesta distribució es pot vore a la Figura 10.

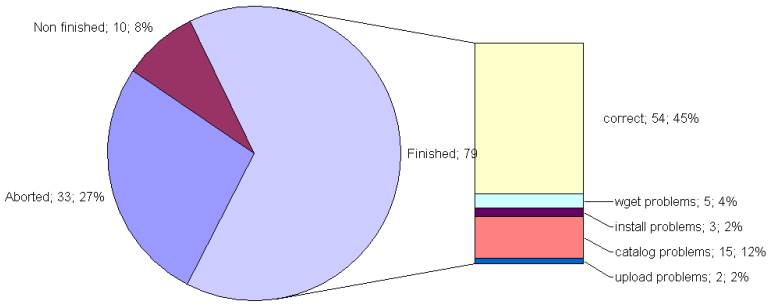

Figura 10: Distribució dels estats dels treballs gLite

Veient els errors obtinguts a l'experiment anterior, calia distingir entre errors, ja que no tots tenien solució. Per això, quedaven automàticament descartats els recursos que fallaven amb *wget* o amb l'execució del programa *blastall*. Aleshores, abans de prendre una decisió, calia veure si es podia solucionar d'alguna manera el problema causant dels errors en la pujada i baixada de fitxers als SEs. La decisió presa va ser la que s'estudiarà al següent apartat (*Selecció de SEs*) i consistia en descentralitzar el sistema d'emmagatzemament, originalment concebut per estar a la màquina local *nephthys.dsic.upv.es*. Amb els canvis proposats, es creaven diverses rèpliques al llarg de tota la infraestructura d'EGEE-II, que augmentava l'accessibilitat als fitxers, i la pujada de fitxers es feia al

SE més proper al CE on es realitzava l'execució, incloent una rèplica a *nephthys.dsic.upv.es*.

Finalment, un cop n'hi ha un subconjunt de recursos de còmput on es pot garantir que els programes desenvolupats van a funcionar apropiadament, cal decidir quants treballs van a llançar-se i fer la divisió entre els recursos disponibles. Sempre que siga possible, la distribució de treballs entre els CEs no serà homogènia sinó que primer s'assignaran treballs als recursos més ràpids, òbviament.

4.2. Selecció de SEs i rèpliques de fitxers

Els treballs *Grid*, com qualsevol altre tipus de treball, necessiten una sèrie de dades d'entrada, que es podrien dividir en paràmetres i fitxers de dades. A alguns dels *middlewares Grid* més emprats (com poden ser *Globus Toolkit*, *LCG-2* o *gLite*) no n'hi ha massa problemes per al pas de paràmetres però els fitxers de dades d'entrada adjunts ja constitueixen un cas diferent. Encara que amb certes limitacions, es poden passar juntament amb els treballs fitxers menuts de dades però molt sovint els fitxers necessaris són excessivament grans, cosa que desaconsella totalment l'ús de l'anomenada safata d'entrada (*input sandbox*).

L'ús de la safata d'entrada d'una forma desmesurada és evident que contribueix a saturar l'ample de banda de la màquina encarregada de fer la submissió de treballs. Açò és degut al fet que les dades incloses en les safates d'entrada i eixida es transfereixen i copien temporalment al Gestor de Tasques (Resource Broker), i d'ací als Computing Elements. A més d'ineficient, és una opció desaconsellable perquè pot saturar l'espai en disc dels gestors de tasques. A més, aquesta saturació seria fàcilment evitable (sobretot en els casos en què es fa un difusió a tots els CEs d'un determinat fitxer, com per exemple una base de dades necessària per a una recerca BLAST) mitjançant l'ús dels sistemes d'emmagatzemament disponibles en infraestructures *Grid* com la d'EGEE-II. L'ús dels SEs en primer lloc aconsegueix superar els problemes d'ample de banda de la interfície d'usuari traslladant-los al propi SE (més preparat per a aqueix tipus de requeriments) i també permet aprofitar el fet que es tracte d'un sistema d'emmagatzemament distribuït amb totes les facilitats corresponents, com ara la creació de rèpliques dels fitxers.

La selecció de SEs i la rèplica de certs fitxers fonamentals per a l'execució d'un treball *Grid* serà una altra de les tasques prioritàries en el desplegament d'un experiment massiu *Grid* en una infraestructura com la de EGEE-II. Com ja s'ha comentat a l'anterior apartat, s'ha de buscar un bon acoblament entre els CEs escollits i els SEs on estaran les distintes rèpliques dels fitxers d'entrada gastats a les execucions dels programes.

D'entrada, hi ha dues estratègies bàsiques pel que fa a la localització dels magatzems contenidors de fitxers d'entrada: centralització (tots els CEs accedeixen al mateix SE) i descentralització (cada CE accedeix al seu propi SE). La primera de les opcions ha estat comentada amb anterioritat i converteix el SE triat en el coll de botella de totes les operacions d'entrada i eixida, mentre que la segona opció, tot i que aconseguiria les millors

prestacions sense dubtes, tindria un impacte negatiu en el global de la infraestructura *Grid* de EGEE-II degut a l'espai de memòria secundària emprat.

Tenint en compte les dues tendències, amb els seus avantatges i inconvenients, es pot optar per una solució de compromís que se situe entre ambdues estratègies. La solució emprada necessitarà d'un estudi per trobar els millors candidats a albergar rèpliques dels distints fitxers. Aquest estudi passarà per elaborar un llistat amb tots els SEs més propers associats als CEs escollits i filtrar aquells que no tinguen les característiques tècniques mínimes aconsellables (per exemple, un ample de banda mínim i una quantitat de memòria física disponible mínima).

Després d'aquest primer filtrat, caldria decidir quin nombre de rèpliques (i per tant de SEs) es volen realitzar, que hauria de dependre del nombre d'elements de còmput emprats a l'experiment i del màxim espai de memòria que es pretén gastar de tota la infraestructura *Grid*. Un cop decidit el nombre, cal plantejar-se la recerca dels hostes de les rèpliques com a representants d'agrupacions de CEs, que bé podrien fer-se en base a criteris de proximitat administrativa i geogràfica. Per això, s'assignaran els SEs de forma progressiva en base al pes relatiu de cada domini dins del conjunt de CEs disponibles, fent que la situació administrativa i geogràfica de les rèpliques siga proporcional a la situació dels elements de còmput. Una de les configuracions escollides en experiments massius seqüencials sobre la organització virtual de biomedicina en EGEE-II seguint els criteris anteriors és la resumida a la Taula 2.

nephthys.dsic.upv.es
clrlcgse01.in2p3.fr
clrauvergridse01.in2p3.fr
mu2.matrix.sara.nl
grim-se.iucc.ac.il
se.keldysh.ru
gridba6.ba.infn.it
prod-se-01.pd.infn.it
eymir.grid.metu.edu.tr
se01.ariagni.hellasgrid.gr
tbn18.nikhef.nl

Taula 2: SEs escollits a l'experiment.

Pel que fa als fitxers d'eixida, el tipus de problemàtica és molt diferent, ja que no té sentit tindre diverses rèpliques d'aquestos fitxers, en tot cas, es podria fer una còpia dels fitxers resultats al SE més proper del recurs de còmput i una rèplica al SE més proper a la interfície d'usuari, per exemple, *nephthys.dsic.upv.es*.

4.3. Especificació del treball

Un cop escollides les màquines on van a llançar-se els treballs i les màquines que seran magatzems de les dades necessàries per a la seua

execució, cal treballar una part fonamental del sistema: el treball pròpiament dit.

En aqueix sentit, caldrà elaborar una especificació en el llenguatge de descripció de treballs en *LCG-2/gLite*, o siga, en JDL (*Job Description Language*). Un exemple de fitxer JDL podria ser el recollit a la Figura 11.

```
Type = "Job";
VirtualOrganisation = "biomed";
Executable = "shell-BiG-seq.sh";
Arguments = "ramses.dsic.upv.es 0 11thExp seqfile290.sqf";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"/home/gaparicio/blast/shell-BiG-seq.sh"};
OutputSandbox = {"std.out","std.err"};
MyProxyServer = "grid001.ct.infn.it" ;
```

Figura 11: Exemple de JDL.

Es tracta d'un exemple de treball seqüencial, ja que en cas de ser un treball paral·lel (MPI) caldria afegir dos paràmetres més, *JobType* i *NodeNumber*, que indicarien el tipus de treball i el nombre de nodes requerits per a l'execució del treball. En aquest darrer cas, *JobType* hauria de tindre el valor "MPICH" i *NodeNumber* el valor de nodes desitjat.

La resta del fitxer JDL seria igual, o siga, començant per un convencional *Type="Job"*, indicant l'organització virtual en què es llança el treball (*VirtualOrganisation*), quin és el fitxer que cal executar tan prompte l'element de còmput estiga lliure (*Executable*), quins són els arguments que s'han de passar a l'executable anterior (*Arguments*), en quins fitxers es dipositaran les eixides estàndard i les eixides d'error (*StdOutput* i *StdError*), quins fitxers cal que es passen conjuntament amb el llançament del treball (*InputSandbox*) i quins fitxers calia recollir a l'hora de prendre l'eixida (*OutputSandbox*) i finalment, quin és el servidor de *proxies* de *MyProxy* (*MyProxyServer*).

Sembla natural que el pes fonamental de l'execució recaurà en l'executable especificat en aquest fitxer i és per això que normalment aquest executable es tracta d'un *shell-script* (habitualment interpretable per *sh* o *bash*) i és dins d'aquest *shell-script* on es desenvolupa tot el treball i es fan les crides a altres executables addicionals i es descarreguen i pugen fitxers des d'un SE o cap a un SE. És per això que el citat *shell-script* pren una importància crucial i convé analitzar el seu contingut, disponible a l'annexe I. Abans d'entrar en matèria, convé comentar que s'empra un *shell-script* i no directament un executable ja que l'executable comporta una sèrie de problemàtiques, com ara resoldre l'impacte dels entorns heterogenis en la compilació dels fonts o haver de fusionar distints processos independents dins d'un mateix executable. D'aquesta manera, treballar amb *shell-scripts* ofereix molta més flexibilitat en el desenvolupament d'aplicacions.

Repassant les línies del *shell-script*, veiem en un principi la recollida dels paràmetres d'entrada, en aquest cas el nom del recurs de còmput, l'identificador del treball *LCG-2/gLite*, l'identificador del experiment i el nom del fitxer de seqüències d'entrada. Aquests paràmetres faciliten el

coneixement, per part del propi treball, de quin paper juga dins de l'experiment global realitzat, estant perfectament identificat. Un cop ja s'han recollit els paràmetres d'entrada, és el moment de preparar l'entorn per a l'execució, és a dir, la construcció de l'estructura de directoris, els canvis pertinents en els fitxers passats per la safata d'entrada (canvis de directori, canvis de permisos, etc.) i l'establiment del directori actual.

Amb els passos anteriors, es dóna per tancades les accions referents als paràmetres i fitxers d'entrada i de preparació de l'entorn. Ara és el moment de començar amb l'execució de les accions específiques del problema concret. Com és habitual en qualsevol treball, primerament es farà una recerca dels fitxers necessaris per a la seua execució (òbviament, fitxers que no s'han passat per la safata d'entrada). Per això, serà habitual l'ús en aquest primer moment de comandaments de descàrrega de fitxers, com ara *wget* per a fitxers a la infraestructura general d'*internet* o *lcg-cp* quan es tracta de fitxers emmagatzemats en elements d'emmagatzematge de la infraestructura *Grid* d'*EGEE*. I un cop descarregat, també és comú tractar-los d'alguna manera, ja siga simplement descomprimint els fitxers, instal·lant el programa o formatejant una base de dades.

Un cop estan disponibles tots els fitxers necessaris per a l'execució, ja es pot procedir a l'execució pròpiament dita. El programa central normalment haurà estat instal·lat a la fase anterior (en el cas d'execucions BLAST seqüencials, serà l'executable *blastall* del *NCBI*) i necessitarà d'una sèrie de paràmetres, alguns comuns a tots els treballs de l'experiment i altres determinats al propi fitxer JDL com a arguments (tal i com s'especifica a la fase inicial del treball). L'objecte d'aquesta execució serà l'obtenció d'unes dades resultats que caldrà empaquetar a l'última fase, triant la millor manera d'oferir aquestes dades a l'usuari.

La fase final consistirà justament en el tractament de les dades resultats, creades a partir de l'execució. Els resultats estaran disponibles mitjançant dos formats principals: fitxers d'eixida i eixides estàndard (eixida i error). Les eixides estàndard estaran disponibles (un cop finalitze el treball) a la safata d'eixida (amb les directives concretes a l'efecte), mentre que els fitxers d'eixida serà habitual que reben algun tractament previ, com ara un filtrat de les dades o una compressió mitjançant *gzip*. Com ja passava amb els fitxers que s'enviaven conjuntament amb el treball a la safata d'entrada, en la fase d'eixida passa exactament igual. Depenent de la grandària dels fitxers d'eixida es podrà optar per obtenir els fitxers amb la safata d'eixida (el cas de fitxers menuts) o buscant alternatives com els elements d'emmagatzematge (el cas dels fitxers grans). Després de resoldre el problema dels fitxers d'eixida ja es pot procedir a abandonar l'element de còmput, sense oblidar-se d'esborrar els fitxers i directoris prèviament creats. De no fer-ho així, només s'estaran deixant fitxers innecessaris sinó que es contribuirà a la desestabilització dels sistemes de còmput emprats, i del sistema en un sentit global. Les bones pràctiques ajuden a l'administració dels sistemes i a mantenir unes bones prestacions en el futur pròxim.

4.4. Automatització del sistema

Abans d'encetar els tests, encara queda construir un sistema automatitzat que porte un control i monitorització de la marxa dels experiments. El sistema s'hauria d'encarregar principalment del llançament dels distints treballs *LCG-2/gLite* en què s'haja dividit l'experiment i de la seua monitorització, rellançament (en cas que siga necessari) i obtenció de resultats.

Aquest sistema ha de ser eficaç, eficient i robust. Com es vorà, moltes vegades serà complicat treballar alhora per aquestes tres característiques fonamentals. L'eficàcia serà la garantia que tard o d'hora el sistema arribarà a un estat en què el treball global haurà finalitzat amb èxit. Per altra banda, l'eficiència garantirà què la distribució de la càrrega del treball és l'òptima i s'arriba a la finalització del treball el més prompte possible, adequant-se el sistema periòdicament a possibles canvis en l'estructura global de la infraestructura *Grid*. Finalment, el sistema ha de ser robust de manera que siga capaç de reprendre una execució interrompuda en qualsevol moment.

Òbviament, són objectius molt ambiciosos que necessitaran d'accions concretes per poder garantir-los i moltes vegades caldrà prendre un compromís quan l'efecte de les accions millore un objectiu però en perjudique un altre. Primerament cal aclarir que garantir l'èxit és d'entrada una fita a la que ningú es pot comprometre i serà necessari assumir que hi haurà una disponibilitat de recursos mínima i una configuració adequada dels recursos.

Un dels punts on es posa de manifest la incapacitat per augmentar alhora eficàcia i eficiència és en l'establiment d'un temps màxim de durada de treball, a partir del qual un treball inacabat cal cancel·lar-lo i rellançar-lo. La seua raó de ser és evitar que hi hagen treballs que es queden en execució per errors al sistema d'informació de *gLite* o simplement per haver-se quedat bloquejats. La cancel·lació de treballs és obvi que és imprescindible per garantir l'eficàcia del sistema, ja que calen mecanismes per desbloquejar les execucions, però també constitueix un risc en tant que és complicat (o fins i tot impossible) distingir entre un treball bloquejat i un treball que evoluciona amb lentitud (tots els recursos no tenen les mateixes prestacions i sovint les aplicacions emprades són heretades i no se'ls pot afegir mecanismes per informar del seu progrés).

La robustesa s'aconseguirà creant un sistema d'informació actualitzat amb una freqüència configurable. Aquest sistema d'informació contindrà les dades necessàries per a restaurar l'execució a pesar de possibles errors ocasionals. Per tant, hi haurà un registre de tots els treballs que componen el treball global, on estarà disponible un identificador del treball, informació sobre l'estat puntual d'aqueix treball i un comptador amb les vegades que un determinat treball ha estat rellançat. Per altra banda, també hi haurà una estructura amb fitxers que contindran els identificadors dels treballs *gLite* que conformen el treball global. Amb les dues estructures es manté l'enllaç amb els treballs *gLite*, que són qui en realitat executen el treball.

4.5. Particionament de dades i distribució de la càrrega.

Un dels passos més importants abans de sotmetre els diferents treballs és el particionament de les dades i la distribució de la càrrega. Com més avant es demostrarà, aquests dos aspectes estan molt estretament relacionats amb la productivitat i les prestacions de l'experiment.

Com es comenta a l'apartat referent a l'automatització del sistema, els treballs han de ser caracteritzats d'acord amb un temps de *wall-time*, a partir del qual l'automatisme descartarà qualsevol treball en execució, entenent que aqueix treball s'ha penjat i no serà productiu. Aquest paràmetre temporal és molt crític, ja que un valor molt menut descartarà molts treballs lents i un valor elevat farà que el sistema espere de forma innecessària i excessiva per a descartar treballs penjats.

No obstant, triar un bon valor per al *wall-time* és un problema complex. En el sistema automatitzat, s'han triat valors relacionats amb un la durada mitjana d'un treball, per exemple, el doble de la durada mitjana. Encara que sembla una estratègia raonable, quan es treballa amb fitxers de seqüències altament heterogènies (pel que fa a la longitud de les seqüències) no és possible garantir que el valor de *wall-time* triat vaja a ser superior al treball més llarg. En el pitjor dels casos, l'experiment mai finalitzarà ja que cap recurs dels disponibles serà capaç d'executar el treball més llarg dintre del temps especificat pel paràmetre *wall-time*.

Per tant, es fa imprescindible homogeneïtzar els costs computacionals dels treballs corresponents a cada bloc, i aquest esforç es pot fer en l'etapa de particionament de dades. La Figura 12 mostra l'evolució del percentatge de treballs finalitzats al llarg del temps amb dues estratègies diferents de particionament de dades (homogeneïtzant el nombre de seqüències en els fitxers dels diferents blocs o homogeneïtzant la grandària dels fitxers d'entrada en el blocs). Aquests experiments sobre metagenomes de microbiota intestinal han utilitzat els mateixos recursos en ambdós experiments i seran analitzats amb més profunditat més endavant a la secció de tests seqüencials.

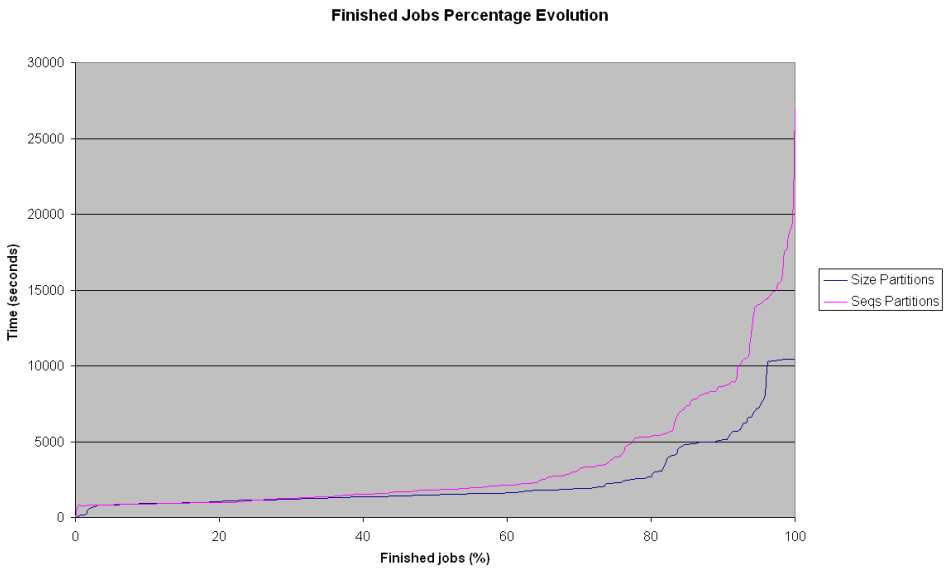

Figura 12: Comparació de les dues estratègies de particionament de dades.

Com es mostra a la darrera figura, amb una estratègia de particionament de dades orientada a la grandària dels fitxers d'entrada, l'experiment arriba a la seua finalització en aproximadament 10.000 minuts (una setmana) i emprant l'estratègia de particionament de dades orientada a les seqüències finalitza aproximadament als 30.500 minuts (tres setmanes). Açò és una clara demostració de com el particionament de dades pot afectar a les prestacions finals. És obvi que no serà senzill determinar quina és la millor estratègia de particionament de dades per a un experiment nou i en la majoria dels casos, la millor solució serà una combinació de diversos criteris, sempre tenint en compte els resultats temporals històrics.

Per altra banda, la distribució de la càrrega és un altre aspecte que cal analitzar per tal d'incrementar les prestacions d'un experiment. No tots els CEs tenen les mateixes prestacions, i la distribució de la càrrega ha de ser realitzada d'acord amb aquestes capacitats. El factor clau és el nombre de treballs concurrents que un CE específic pot executar alhora. Aquest valor estarà estretament relacionat amb el temps de cua i la velocitat del processador (els bytes d'una seqüència d'entrada que poden ser processats per unitat de temps), i els treballs han de ser assignats de forma asimètrica d'acord amb aquest nombre de treballs màxims concurrents.

4.6. Serveis *Grid*

Per tal d'accedir als diferents sistemes implementats és necessari un punt d'accés des del qual l'usuari tindrà un accés ordenat i limitat als recursos i serveis del sistema. Aquest punt d'entrada s'ha implementat mitjançant l'ús de Serveis *Grid*, amplament comentats a la secció corresponent de l'estat de l'art. Mitjançant el Serveis *Grid* s'afegeix al sistema global diverses característiques que seran útils, com es podrà veure més endavant.

Les principals funcions dels serveis *Grid* implementats seran:

- Establir una interfície senzilla basada en estàndards.

- Recollir tots els treballs proposats pels clients.
- Monitoritzar els treballs en execució.
- Servir els resultats als clients.
- Independitzar la part client de la infraestructura *Grid*.
- Fer còpies de seguretat periòdiques per poder restaurar l'estat dels treballs en qualsevol moment.
- Aportar mecanismes de seguretat, basats en SSL, per tal de garantir la integritat de les dades i respectar la privacitat i els nivells d'autorització de cada usuari.

Cal fer notar que, per mantenir l'afany estandarditzador, s'optarà per interfícies similars a les dels equivalents executables seqüencials, com ara les del *blastall* del *NCBI*. Així s'aconsegueix que l'impacte sofert per l'addició d'una capa *Grid* siga el mínim visible per part de l'usuari. Una bona prova d'això ha sigut l'adaptació de Blast2GO per a suportar, a més del servei de NCBI, el servei *Grid* d'este treball.

5. EXPERIMENTS DESENVOLUPATS

Després del desplegament dels sistemes d'execució massiva a les infraestructures *Grid* d'EGEE i EELA, arriba l'hora de fer les proves i execucions de diferents casos d'estudi. Aquestes proves es faran en base a les dues aproximacions anteriorment descrites, segons estiguen basades en ferramentes seqüencials o paral·leles. Caldrà justificar adequadament l'ús i conveniència de cadascuna de les dues estratègies, així com posar de relleu els principals problemes a l'hora d'emprar-les.

La diferència entre ambdues aproximacions sembla que haja de ser simplement el tipus d'executable i el nombre de recursos que l'executaran, però es podrà comprovar que hi ha molts més aspectes que aniran apareixent i condicionaran la decisió de gastar un sistema o l'altre.

Les proves s'han fet amb distintes ferramentes biomèdiques, com poden ser BLAST, MrBayes o CLUSTAL, tant en les seues versions seqüencials com les versions paral·leles basades en MPI. Els resultats se centraran en execucions per a l'alineament de seqüències de proteïnes i nucleòtids. Per això es prendrà com a base l'algorisme BLAST i els distints executables implementats pel *NCBI*, tant la seua versió seqüencial (*blastall*) com la versió paral·lela en MPI (*mpiblast*). Centrar-se en l'algorisme BLAST està justificat donades les pròpies característiques de l'algorisme, que el fan potencialment escalable, ja que és possible en tot moment dividir un treball global en subtreballs, ja siga dividint el fitxer de seqüències d'entrada en blocs o fent el mateix amb la base de dades on es fa la recerca.

Per altra banda, el cas amb les ferramentes filogenètiques (MrBayes i CLUSTAL) no tenen un comportament tan escalable, ja que els propis algorismes necessiten de totes les seqüències d'entrada per a obtenir el resultat desitjat (no es poden particionar) i tampoc n'hi ha cap base de dades sobre la qual es fa una recerca. Aleshores, és obvi que es tractaran d'algorismes amb més comunicacions, cosa que limitarà l'escalabilitat de l'algorisme. A més, en el cas de MrBayes, en la seua versió 3.1.2 hi ha una limitació del nombre màxim de nodes a la versió paral·lela, que ve condicionada pel producte del nombre d'execucions *nruns* i el nombre de cadenes de Markov *nchains* i tenen per valors per defecte el de 2 execucions i 4 cadenes de Markov. En el cas de CLUSTAL, la limitació està en funció del nombre de seqüències, cosa que fa aquest algorisme molt més versàtil. El problema és que, tot i ser ambdós algorismes filogenètics, la tècnica emprada és diferent, la precisió dels resultats i les característiques també són diferents i, per tant, el seu àmbit d'aplicació també és diferent.

5.1. Experiments *Grid* seqüencials

La primera aproximació és la seqüencial, on es gastarà com a executable base el *blastall* del *NCBI*. És en principi l'aproximació amb menys

requeriments i la que millor s'adaptarà als requeriments de les infraestructures *Grid* emprades, ja que encara no n'hi ha encara un suport generalitzat a la computació paral·lela. Els dos experiments presentats són dos grans alineaments de dos metagenomes d'interés biològic: un del mar dels Sargassos i una bateria de metagenomes de microbiota intestinal. Un metagenoma és una col·lecció de gens que pot ser analitzat com si es tractara d'un sol gen, de manera que és una pràctica d'anàlisi molt més econòmica ja que no cal ni aïllar els gens ni cultivar-los a un laboratori. En concret, l'anàlisi d'un metagenoma engloba més fases que no el simple alineament de seqüències amb BLAST, com podria ser el filtrat de la seua eixida o la creació d'arbres filogenètics.

A continuació es descriuen breuement els metagenomes utilitzats com a bateria de test, justificant el seu interés, a més dels resultats obtinguts en els experiments.

5.1.1. Metagenoma del mar dels sargassos

El primer experiment seqüencial s'ha desenvolupat amb el processament del metagenoma del mar dels sargassos, comentat en el capítol d'estat de l'art. Sobre aquest metagenoma es farà una recerca a la base de dades pública "*Non-Redundant*", amb la particularitat que s'han filtrat totes les entrades amb eukaryotes, ja que és conegut que no hi ha cap cèl·lula eukaryota en la mostra. Altres paràmetres destacats de l'execució són el nombre de hits (5.000), el *expected value* (fixat a 0,01) i l'algorisme emprat (*blastx*).

Per tal de processar-lo a la infraestructura *Grid* d'EGEE, es va haver de dividir en 610 fitxers d'aproximadament 1.330 seqüències cadascun, donant lloc a 610 treballs *gLite*, llançats sobre 122 CEs, amb una proporció de 5 treballs per CE. Òbviament, caldrà fer rellançaments en cas d'errades i en cas que un treball tarde en excés (un criteri emprat als experiments i que ha donat bon resultat seria considerar que un treball està tardant massa quan ha superat el doble del temps mitjà d'un treball en experiments anteriors). L'experiment consistirà en l'execució de cada treball de forma independent a la resta.

A la Taula 3 apareix un resum amb els principals paràmetres del metagenoma del mar dels sargassos codificat. Es pot apreciar tant la grandària en bytes del fitxer, com el nombre de seqüències, la mitjana de bytes per seqüència, la desviació típica de la distribució de bytes per seqüència i el nombre de blocs emprats a l'experiment. Com es comprovarà al següent experiment, a pesar que les mitjanes de bytes per seqüència són semblants, la desviació típica és molt inferior, cosa que fa que un particionament per seqüències (en ser les seqüències gairebé de la mateixa longitud) no desequilibre la càrrega en excés.

Fitxer	Grandària (Mbytes)	Nombre de Seqüències	Nombre de Blocs	Mitjana Seqs. per Bloc	Desv. Seqs. per Bloc
Sargasso_Sea.fasta	881	811.372	610	1.091	481

Taula 3: Resum dels principals paràmetres del metagenoma del mar dels sargassos.

En aquest cas, s'ha optat per no fer cap filtrat previ de CEs i adaptar-se de forma dinàmica a l'estat dels diferents recursos emprats, tot seguint una tècnica de llistes negres i blanques, amb certes optimitzacions encarades a millorar les prestacions globals del sistema. La idea bàsica és identificar els recursos que causen errors crítics de forma reiterada i descartar-los per a futur llançaments. Paral·lelament, cal identificar també els recursos que funcionen correctament, prenent nota de les seues prestacions per emprar-los de la millor manera en el futur.

Amb la tècnica de llistes blanques i negres, l'experiment ha finalitzat amb un subconjunt de 62 CEs productius, que suposa un 51% dels 122 CEs inicials. Podria semblar una xifra menuda de recursos, però pràcticament dobla el nombre de recursos que en un moment determinat funcionen sense problemes. Açò és una clara demostració que l'estat dels recursos dins de la infraestructura EGEE és molt dinàmica i el fet que un treball acabe amb èxit o amb error en un determinat recurs no garanteix que el proper treball llançat al mateix recurs, acabe de la mateixa manera.

Pel que fa al comportament temporal de l'experiment, a la Figura 13 es pot apreciar la evolució en el percentatge de treballs finalitzats amb èxit al llarg del temps.

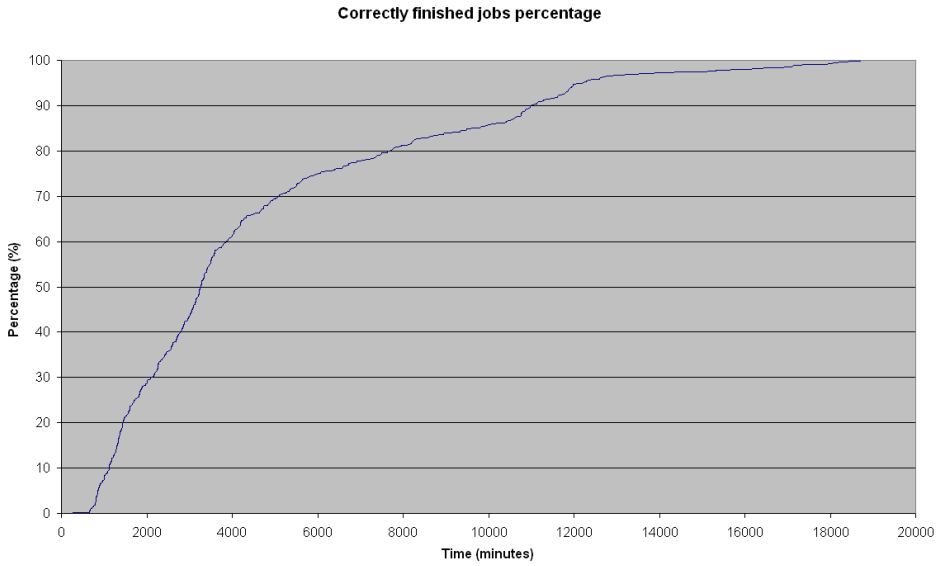

Figura 13: Evolució en el percentatge de treballs finalitzats amb èxit a l'experiment del metagenoma del mar dels sargassos.

Lògicament, hi ha una etapa inicial on no s'obté cap resultat i el temps passa mentre que el percentatge de treballs finalitzats amb èxit roman a zero. A partir dels 800 minuts ja comencen a finalitzar els treballs més ràpids, cosa que fa que la corba passe en un espai de temps reduït (dels 800 minuts als 5000 minuts, aproximadament) del 0% al 70%. A partir del 70%, el ritme es fa més lent, degut a treballs que han fallat i s'han hagut de rellançar a altres recursos. Per això, s'inverteix el triple de temps en el 30% final que en el 70% inicial. Aquesta dada podria ser preocupant si no fora perquè es tracta d'una fase inicial d'una anàlisi amb més etapes, com és l'anàlisi d'un metagenoma. La corba, en cas d'una anàlisi d'un metagenoma pot ajudar a dosificar la càrrega d'una infraestructura *Grid*

gran però amb recursos finits i amb limitacions, però també hi ha diferents estratègies per combatre-la en cas que fora necessari.

A la Figura 14 es pot apreciar la velocitat en que es van processant les seqüències d'entrada. Es veu clarament que s'adquireix una velocitat punta al voltant de les 7.000 seqüències processades per hora, però degut a que no hi ha aplicada cap estratègia per evitar-ho, la velocitat va disminuint progressivament fins al final de l'experiment.

Sovint no es consideraran solucions concretes per evitar les corbes finals perquè l'alineament de seqüències serà en la majoria d'anàlisis metagenòmiques una fase inicial que serà post-processada i pot donar lloc a altres fases com la de creació d'arbres filogenètics. En cas que l'alineament de seqüències fora l'objectiu de l'anàlisi, caldria optar per tècniques com el rellançament replicat de treballs que han fallat (o de tots en general, augmentant el factor de replicació a mesura que queden recursos lliures dels inicialment assignats) i anar cancel·lant els treballs en execució si una rèplica seua ja ha donat resultats. Si hi ha més fases, l'efecte de la corba no només es dilueix sinó que pot ajudar a distribuir la càrrega global d'una forma progressiva.

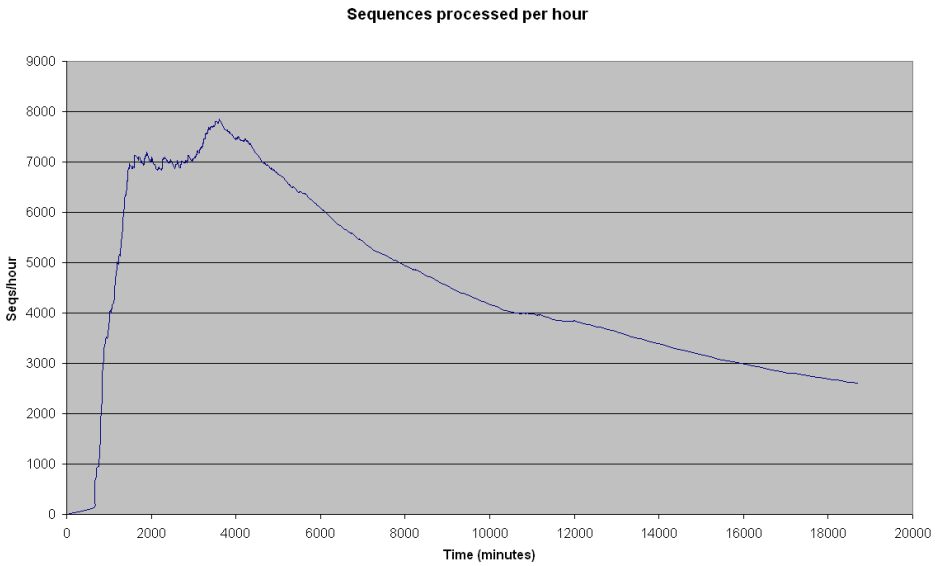

Figura 14: Evolució de les seqüències processades per hora en l'experiment del metagenoma del mar dels Sargassos.

L'experiment tarda un total de 302 hores (13 dies) en finalitzar, mentre que el sumatori d'hores invertides en cadascun dels treballs que componen l'experiment és de 12.271 hores (512 dies). Això es tradueix en una acceleració respecte del treball seqüencial de 39 en el moment de finalització de l'experiment, però pràcticament del doble (66) si es calculara per al moment en què s'arriba a la finalització del 90% dels treballs. Açò és degut a la penalització dels treballs pendents sobre el sistema i ja ha estat anteriorment comentat.

A banda del comportament temporal de l'experiment, es pot parlar de més coses, com ara els treballs *gLite* llançats en total, que sumen 1.291 treballs o, el que és el mateix, un percentatge del 112% de rellançaments. Aquesta dada és totalment congruent amb el fet que només 62 elements de còmput (el 51%) han estat capaços de produir resultats i amb altres causes, com

ara l'errada esporàdica de recursos productius o problemes amb l'ajustament del temps llindar, a partir del qual un treball és descartat per haver durat massa temps.

Els resultats generats han estat distribuïts en tants fitxers com blocs (610), sumant un total de 14,3 Gbytes comprimits. Això significa que cada bloc ha generat una mitjana de 22 Mbytes (amb una desviació típica de 11,6 Mbytes) i cada seqüència uns 18,5 Kbytes.

5.1.2. Metagenomes de microbiota intestinal en humans

El segon experiment seqüencial es farà sobre el conjunt format pels tretze metagenomes de microbiota intestinal humana descrit a l'apartat anterior d'estat de l'art. Amb el mateix conjunt de dades, es realitzaran dos experiments diferents atenent a dues estratègies de particionament diferent, un particionament per seqüències (cada bloc té un mateix nombre de seqüències) i un particionament per grandària (cada bloc té una grandària en bytes aproximadament igual). Els experiments es realitzaran sobre 13 fitxers corresponents a 13 metagenomes de microbiota intestinal publicats al Japó de persones de diferents sexes i edats, barrejant gent malalta i gent sana. A la Taula 4 es pot vore un resum amb les característiques més interessants de cada metagenoma, com són el nom del fitxer on està codificat el metagenoma, la grandària en bytes del metagenoma, les seqüències que el conformen, la mitjana de bytes per seqüència i la desviació típica, per tenir dades objectives sobre la distribució de seqüències i la variabilitat de la seua longitud, cosa que marcarà el particionament de les dades i les prestacions finals de l'experiment. Per això, els últims camps són el nombre de blocs per metagenoma per a les dues estratègies de particionament, primer(1) el particionament per seqüències i segon(2) el particionament per grandària.

Fitxer	Grandària	Seqüències	Mitjana	Desv.	Blocs[1]	Blocs[2]
BAAU.fasta	40.681.594	28.900	1.315	951	22	22
BAAV.fasta	46.540.773	36.326	1.190	764	27	25
BAAW.fasta	25.937.934	16.535	1.473	1.884	13	14
BAAX.fasta	49.241.036	36.455	1.259	671	27	26
BAAY.fasta	42.872.747	30.198	1.327	996	23	23
BAAZ.fasta	41.924.237	31.237	1.250	1.023	23	22
BABA.fasta	48.718.080	35.177	1.293	757	26	26
BABB.fasta	31.210.347	20.226	1.448	1.692	15	17
BABC.fasta	15.442.608	9.958	1.455	2.016	8	8
BABD.fasta	49.799.460	37.296	1.244	998	28	26
BABE.fasta	29.109.717	20.532	1.325	1.262	15	16
BABF.fasta	27.513.565	16.164	1.604	2.028	12	15
BABG.fasta	46.651.330	34.797	1.249	691	26	25
Totals	495.643.428	353.801	1.308	1.140	265	265

Taula 4: Resum dels principals paràmetres dels metagenomes de microbiota intestinal, cada fila correspon a una mostra d'una persona diferent.

Els particionaments s'han realitzat sense barrejar els fitxers d'entrada, per a una millor gestió posterior dels fitxers de resultats. Per tal de poder

apreciar millor com afecta l'estratègia de particionament emprada, s'ha optat per tindre el mateix nombre de blocs totals, però distribuïts de diferent manera. El particionament per seqüències és el mateix que en el cas de l'experiment del metagenoma del mar dels Sargassos. La novetat és el particionament per grandària (o caràcters), és a dir, el que es persegueix no és que tots els fitxers tinguen un nombre semblant de seqüències sinó un nombre semblant de bytes. Quan les seqüències d'entrada són d'una longitud semblant i poc variable es pot fer el particionament per seqüències sense el risc de fer una distribució descompensada de la càrrega. El problema apareix quan es treballa amb fitxers de seqüències amb una variabilitat en la seua longitud molt elevada, ja que el temps de còmput és proporcional a la longitud de les seqüències d'entrada. En aquest darrer cas es fa imprescindible canviar el mètode de particionament i optar per un particionament orientat a equilibrar el nombre de caràcters dels fitxers, tenint en compte que una seqüència no es pot partir en dos trossos. Les particions es van fer d'un màxim de 2 Mbytes, sumant un total de 265 particions, 13 d'elles més menudes per ser les seqüències sobrants de cadascun dels metagenomes.

Un cop fet el particionament, arriba el moment de fer la distribució dels treballs entre els CEs escollits (un total de 52, exactament els que en aquell precís instant eren productius) de manera que s'envien més treballs a aquells CEs que han estat més productius en experiments anteriors i menys als que no ho han estat tant. Òbviament, es tractarà d'afegir un factor de correcció i no aplicar proporcionalitat, ja que en cas contrari es podria contribuir a saturar els millors recursos i restar l'eficiència al sistema que s'estava intentant buscar. També caldrà fer rellançaments en cas d'errades i en cas que un treball tarde en excés, com ja es feia a l'experiment anterior. L'experiment consistirà en l'execució de cada treball de forma independent a la resta, fent una recerca a una base de dades creada a partir dels tretze metagenomes de microbiota intestinal humana, ja que es pretén fer un estudi de recerca de similituds i diferències entre la microbiota intestinal dels diferents individus. Altres paràmetres destacats de l'execució són el nombre de hits (20.000), el *expected value* (fixat a 0,01) i l'algorisme emprat (*tblastx*), que és teòricament 6 vegades més costós computacionalment que *blastx* (ja que cada nucleòtid de la base de dades es tradueix en sis seqüències per a comparar.

A continuació es comparen i analitzen els resultats obtinguts en ambdós aproximacions.

- Particionament per seqüències

  A la Figura 15 es pot apreciar l'evolució en el percentatge de treballs finalitzats amb èxit amb aquest tipus de particionament.

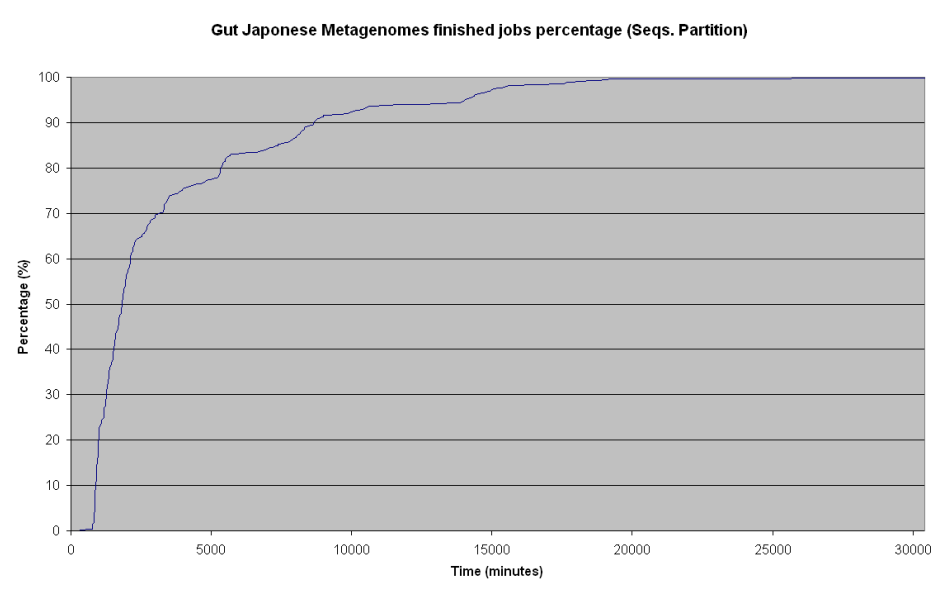

Figura 15: Evolució en el percentatge de treballs finalitzats amb èxit a l'experiment dels metagenomes de microbiota intestinal en humans amb particionament per seqüències.

Com passava a l'experiment anterior, a partir dels 800 minuts ja comencen a finalitzar la resta de treballs, cosa que fa que la corba passe en un espai de temps reduït (dels 800 minuts als 3.000 minuts, aproximadament) del 0% al 70%. Açò suposa un ritme de creixement molt més elevat que abans. A partir del 70%, el ritme es fa més lent, degut a treballs que han fallat i s'han hagut de rellançar a altres recursos. Per això, s'inverteix nou vegades més temps en el 30% final que en el 70% inicial. Novament, no és un fet preocupant ja que es tracta d'una fase inicial d'una anàlisi amb més etapes, com és l'anàlisi d'un metagenoma, i contribuir a dosificar la càrrega a un *Grid* amb recursos finits i amb limitacions.

A la Figura 16 es pot apreciar l'evolució de les seqüències processades per hora en el present experiment. En aquest cas, la velocitat punta sembla ser un inferior (sobre les 6.000 seqüències processades per hora) i també va disminuint progressivament fins a la fi de l'experiment per les causes ja esmentades anteriorment.

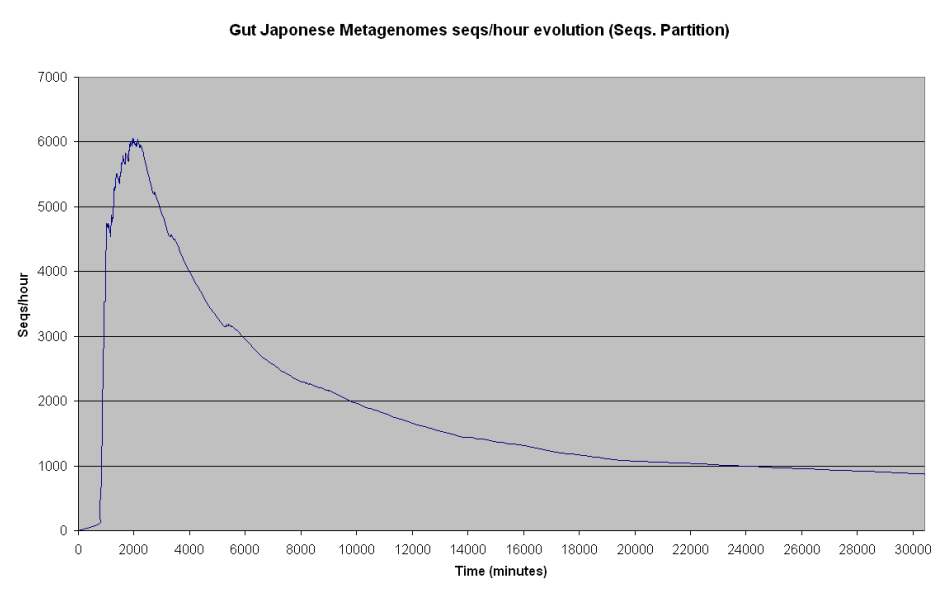

Figura 16: Evolució de les seqüències processades per hora en l'experiment dels metagenomes de microbiota intestinal humana amb particionament per seqüències.

L'experiment tardà un total de 507 hores (21 dies i 3 hores) en finalitzar, mentre que el sumatori d'hores invertides en cadascun dels treballs que componen l'experiment fou de 5.687 hores (237 dies). Això es tradueix en una acceleració respecte del treball seqüencial de 11,2 en el moment de finalització de l'experiment, però més del triple (34,1) si es calculara per al moment en què s'arriba a la finalització del 90% dels treballs, més de quatre vegades si ens centrem en el 80% (50,5) i un màxim de 79,6 al 70%.

Deixant de banda les qüestions temporals, els treballs *gLite* llançats en són 892, és a dir, un percentatge del 236,6% de rellançaments, dels quals la majoria (184,5%) són treballs cancel·lats per haver exhaurit el temps màxim per treball estipulat a l'automatisme. L'augment en els percentatges de rellançament (més del doble que en el cas del metagenoma del mar dels sargassos) i sobretot el gran nombre de treballs cancel·lats es deu fonamentalment a la variabilitat en les longituds de les distintes seqüències d'entrada, cosa que està estretament relacionada amb el temps de durada dels treballs. Per això, els treballs són més llargs i només acaben abans de véncer el temps de *walltime* configurat a l'automatisme en certs recursos molt ràpids.

- Particionament per caràcter.

Pel que fa referència al comportament temporal de l'experiment, a la Figura 17 es pot apreciar l'evolució en el percentatge de treballs finalitzats amb èxit a l'experiment dels metagenomes de microbiota intestinal humans.

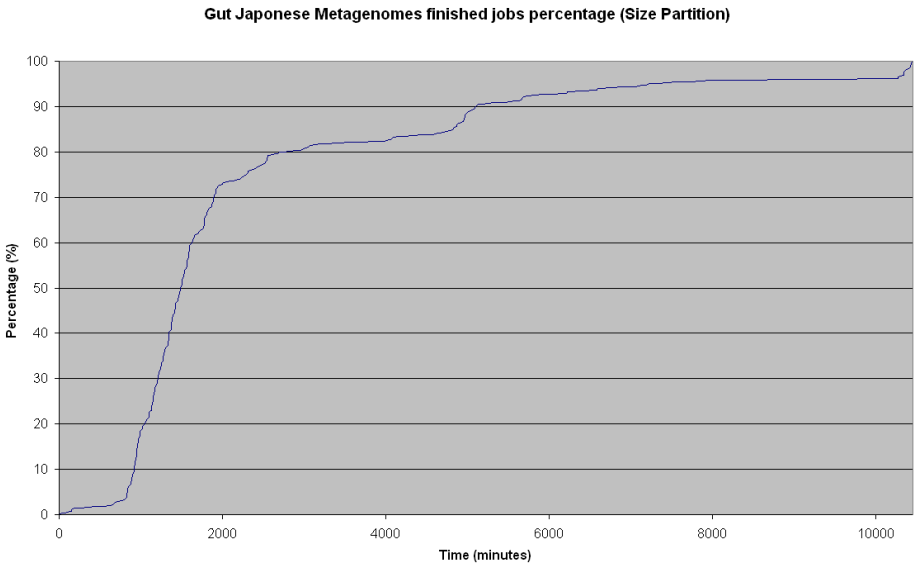

Figura 17: Evolució en el percentatge de treballs finalitzats amb èxit a l'experiment dels metagenomes de microbiota intestinal en humans amb particionament per grandària.

La tendència és molt similar a la de l'experiment anterior amb algunes diferències que caldrà destacar. A diferència de com passava a l'experiment anterior, des del principi de ja hi ha resultats (ja que ara hi ha fitxers d'entrada molt més menuts que abans). Com passava a l'experiment anterior, a partir dels 800 minuts ja comencen a finalitzar la resta de treballs, cosa que fa que la corba passe en un espai de temps reduït (dels 800 minuts als 2.000 minuts, aproximadament) del 0% al 70%. Açò suposa un ritme de creixement molt més elevat que abans. A partir del 70%, el ritme es fa més lent, degut a treballs que han fallat i s'han hagut de rellançar a altres recursos. Per això, s'inverteix cinc vegades més temps en el 30% final que en el 70% inicial. Novament, no és un fet preocupant ja que es tracta d'una fase inicial d'una anàlisi amb més etapes, com és l'anàlisi d'un metagenoma, i contribuir a dosificar la càrrega a un *Grid* amb recursos finits i amb limitacions.

A la Figura 18 es pot apreciar l'evolució de les seqüències processades per hora en el present experiment. Al contrari que passava amb l'experiment del metagenoma del mar dels sargassos, en un principi també hi ha fluctuacions en la velocitat (tal i com passava a l'evolució percentual). En aquest cas, la velocitat punta sembla ser un poc superior (gairebé 8.000 seqüències processades per hora) però també va disminuint progressivament fins a la fi de l'experiment per les causes ja esmentades anteriorment.

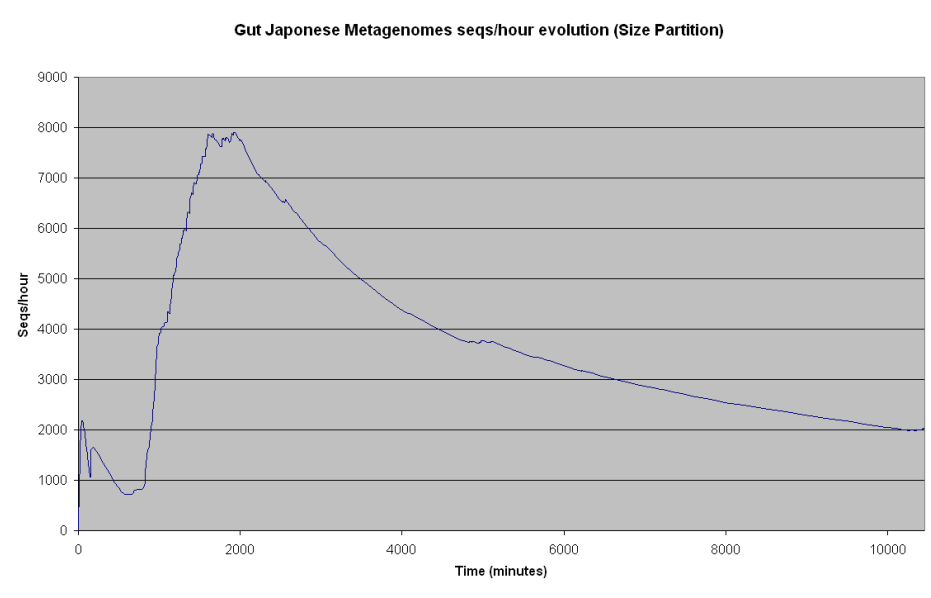

Figura 18: Evolució de les seqüències processades per hora en l'experiment dels metagenomes de microbiota intestinal humana amb particionament per grandària.

L'experiment tarda un total de 174 hores (7 dies i 6 hores) en finalitzar, mentre que el sumatori d'hores invertides en cadascun dels treballs que componen l'experiment és de 5.687 hores (237 dies). Això es tradueix en una acceleració respecte del treball seqüencial de 32,6 en el moment de finalització de l'experiment, però més del doble (60) si es calculara per al moment en què s'arriba a la finalització del 90% dels treballs i gairebé quatre vegades més si ens centrem en el 80% (101) o un màxim de 120 al 70%. Açò és degut a la penalització dels treballs pendents sobre el sistema i ja ha estat anteriorment comentat.

Deixant de banda les qüestions temporals, els treballs *gLite* llançats en en són 421, és a dir, un percentatge del 58,8% de rellançaments, dels quals només un 2,3% són treballs cancel·lats per haver exhaurit el temps màxim per treball estipulat a l'automatisme. La reducció en els percentatges de rellançament (la meitat que l'anterior) i sobretot amb els treballs cancel·lats es deu fonamentalment a la tècnica de particionament per caràcter, que ja s'ha comentat amb anterioritat.

- Resultats i comparativa dels experiments.

Els resultats de l'experiment han estat un total de 265 fitxers comprimits en *gzip* que sumen un total de 5,95 Gbytes amb una mitjana de 23,1 Mbytes per bloc i 17,6 Kbytes generats per seqüència.

Les principals diferències es troben quan es calcula la desviació típica de les grandàries dels resultats per a cada bloc. En el cas de l'experiment amb el particionament per seqüències, la desviació típica és de 17,7 Mbytes, mentre que amb el particionament per caràcters és de 7,6 Mbytes. Aquesta diferència entre les desviacions típiques indica que amb el particionament per caràcters s'ha aconseguit homogeneïtzar molt més els treballs que conformen l'experiment, ja que hi ha una dependència pràcticament lineal entre el cost computacional i la longitud de les seqüències.

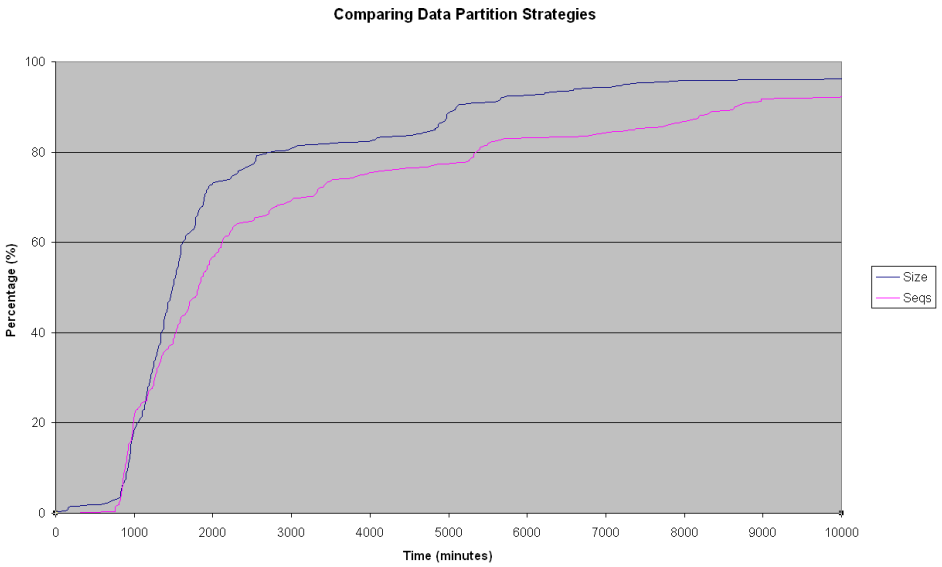

Figura 19: Comparativa d'estratègies de particionament de dades segons l'evolució percentual dels treballs finalitzats dels metagenomes de microbiota intestinal humana.

A la Figura 19 es pot comparar l'evolució percentual de treballs finalitzats en variar l'estratègia de particionament. L'evolució completa es pot vore a figures anteriors i en aquesta ocasió només es veuen els primers 10.000 minuts. Analitzant els 1.000 minuts inicials (més o menys, la durada mitjana per treball que és 1.288 minuts) es pot vore com inicialment l'estratègia per caràcter ofereix més més resultats (ja que té treballs molt menuts degut a les parts sobrants dels distints metagenomes a l'hora de fer el particionament).

Més endavant, l'estratègia per seqüències es recupera i fins als 1.000 té un comportament més ràpid. Açò està relacionat amb la distribució de la càrrega computacional entre els distints treballs que conformen l'experiment. En el cas de particionament per caràcters la distribució és més homogènia i en el cas del particionament per seqüències és més heterogènia. Per això, hi ha un alt nombre de treballs en el cas de particionament per seqüències que són més curts que en l'altre cas (que són pràcticament tots idèntics).

A partir dels 1.000 minuts i fins als 2.000 minuts, mentre que els treballs particionats per caràcter continuen donant resultats a un pas constant, els treballs particionats per seqüències deixen d'oferir resultats (ja que els treballs curts ja han acabat, aproximadament el 50%, i els llargs tardaran molt més). A partir dels 2.000 minuts, el temps de *walltime* ha vençut i el nombre de treballs rellançats mitjançant el particionament per caràcter és molt menor que en el cas per seqüències i això es va repetint al llarg dels distints rellançaments necessaris per a la completa finalització dels experiments.

5.2. Experiments *Grid* amb MPI

En capítols anteriors es fa esment de les possibilitats de fer aproximacions paral·leles per a entorns de computació amb memòria distribuïda. Aquest és el cas de les versions MPI de CLUSTAL-W, Mr.Bayes o BLAST. En aquest sub-apartat es parlarà d'una comparativa entre dos versions BLAST implementades pel NCBI: la seqüencial *blastall* i la paral·lela *mpiblast*, en les seues respectives darreres versions. Tenint en compte els problemes d'infraestructura ja comentats abans, cal veure quin seria el resultat comparatiu en cas que les anteriors problemàtiques estigueren superades i així encoratjar a millorar la infraestructura o deixar d'invertir esforços en aqueix sentit.

La comparativa es realitzarà sobre 2.000 seqüències que pertanyen a un metagenoma de microbiota intestinal. En el cas seqüencial hi haurà un total de 20 treballs independents particionats per caràcters (amb una mitjana de 100 seqüències per treball), intentant que el cost computacional de cada treball siga el més homogeni possible. En el cas paral·lel hi ha un sol treball paral·lel MPI gastant el fitxer original sense particionar i amb un requeriment de 20 nodes. Ambdós experiments es realitzaran gastant com a *Computing Element* el recurs identificat per *ramses.dsic.upv.es:2119/jobmanager-pbs-biomedg* i gastant com a base de dades el conjunt de metagenomes de microbiota intestinal humà també emprat com a base de dades als experiments del sub-capítol anterior.

A la Taula 5 hi ha un resum amb les característiques del fitxer de seqüències emprat als presents experiments, on s'aprecia una desviació típica molt acusada tenint en compte la mitjana de caràcters per seqüència, cosa que aconsellava el tipus de particionament emprat per al cas seqüencial.

Fitxer	Grandària	Seqüències	Mitjana	Desv.	Blocs
human_gut.fasta	7.860.282	2.000	3.797	2.277	20

Taula 5: Resum dels principals paràmetres del fitxer de seqüències de test.

- Cas seqüencial.

  El primer cas realitzat consta de 20 experiments seqüencials, llançats a través del *middleware gLite 3.0*, que tarden una mitjana de 322 minuts i 50 segons amb una desviació típica de 13 minuts i 53 segons, cosa que dóna fe de l'encert en l'estratègia triada en el particionament de dades. L'experiment ha tardat globalment un total de 5 hores i 54 minuts, mentre que si els experiments s'hagueren executat un darrere de l'altre haurien tardat 4 dies, 11 hores i 40 minuts. La poca diferència entre la mitjana dels treballs i la duració global és deguda a la fiabilitat dels recursos on s'han llançat els treballs, ja que són recursos propis i han estat configurats de forma òptima per als experiments realitzats, evitant així els inconvenients trobats quan els treballs es llancen a múltiples recursos de diverses organitzacions.

  Un altre factor a tindre en compte és que l'espai de memòria secundària emprat ve marcat per les 20 descàrregues de la base de dades, que ocupa 473 Mbytes sense comprimir. Com que abans de començar amb el procés és necessari el format de la base de dades, hi

ha un espai de temps en què és necessari que estiga present alhora la versió descomprimida i la formatada (aproximadament de la mateixa grandària) i fent un total de 946 Mbytes per 20 nodes, o siga, un total de 18,4 Gbytes. A més, cal sumar el software necessari per als 20 processos (16,3 Mbytes comprimits per node que es transforma en 325 Mbytes per node quan s'instal·la, o siga 6,3 Gbytes en total) i 7 Mbytes en concepte de seqüencies d'entrada. És a dir, l'espai total requerit és de 24,7 Gbytes, que pot estar a un directori compartit o distribuït entre els diferents *Worker Nodes*.

També es pot fer l'anàlisi de la memòria primària requerida per node. Per a calcular-la, anem a suposar que totes les dades necessàries estan a la memòria. És a dir, la base de dades amb 473 Mbytes, la porció corresponent de les seqüències d'entrada amb 0,4 Mbytes de fitxer d'entrada.

Finalment, fer un breu comentari sobre la quantitat de bytes descarregats i l'ample de banda de la xarxa del *Computing Element* emprat. Com que cal baixar concurrentment 20 versions de tots els fitxers, com per exemple les bases de dades, l'ample de banda ha de ser dividit entre 20 per nodrir totes les descàrregues concurrents. L'impacte sobre el rendiment del sistema no és molt previsible a priori, ja que dependrà de l'ample de banda del servidor on està allotjat el fitxer a descarregar i de l'ample de banda del client, en este cas el *Worker Node*, que és qui demanda el fitxer. El que sí és evident és que l'ample de banda requerit, per tal que el temps de descàrrega de l'experiment global fora el mateix que el d'un experiment individual, és de l'ordre de 20 vegades més gran en el cas en el qual el *coll de botella* de la capacitat d'ample de banda estiga al client i no al servidor, com sembla ser previsible. En l'experiment, els bytes descarregats estarien al voltant de 3 Gbytes, sumant la base de dades comprimida, el software necessari comprimit i les seqüències d'entrada comprimides.

- Cas paral·lel.

Per tal d'equilibrar la càrrega computacional respecte a l'anterior grup d'experiments, el fitxer de seqüències és la suma dels fitxers de seqüències de tots els sub-treballs anteriors. Així s'ha conformat un treball sol treball paral·lel amb el *middleware gLite 3.0,* posant com a requeriment l'ús computacional de 20 nodes, cadascun d'ells amb les mateixes prestacions que els 20 gastats al cas anterior. El temps del treball, que en aquest cas coincideix amb el temps global de l'experiment, és de 6 hores i 26 minuts. Resulta obvi que el temps és pitjor que en el cas anterior, i l'explicació es troba en el tipus de particionament realitzat. Contràriament al que passava en el cas seqüencial, en aquest cas és necessari un post-procés dels resultats oferts per cada procés MPI per tal d'ajustar-se als criteris de recerca desitjats per l'usuari. Per altra banda, els bytes descarregats en aquest experiment són 20 vegades inferiors, ja que només cal baixar una vegada les dades requerides, en lloc de les 20 anteriors. Després hi ha una part comuna amb la part seqüencial, que no és paral·lelitzable, com ara la part del formatat de la base de dades.

Pel que fa a la memòria secundària emprada, només és necessari tindre un espai aproximat de dues vegades la grandària de la base de dades, o siga, uns 946 Mbytes sense comprimir que després del formateig es quedaran en només 473 Mbytes per a les recerques BLAST. També cal sumar uns 325 Mbytes en concepte de software requerit i ja instal·lat, més 7 Mbytes del fitxer de seqüències d'entrada. En total, 1268 Mbytes.

Respecte a la memòria principal requerida, caldrà situar en memòria la vintena part de la base de dades (al voltant 24 Mbytes) i el fitxer de seqüències, que té una grandària de 7 Mbytes descomprimit. En total, 31 Mbytes.

Igual que passa amb la memòria secundària, el nombre de bytes que s'han de descarregar són justament la vintena part que en el cas anterior, ja que només cal baixar la base de dades una sola vegada. De la mateixa manera, l'ample de banda requerit també és menor si es vol mantindre constant el temps de descàrrega.

- Comparativa dels casos seqüencial i paral·lel.

Entre els casos seqüencials i paral·lel hi ha vàries diferències, com s'ha pogut comprovar numèricament. La primera diferència és la referent als temps de processament i després altres coses molt importants com l'espai en disc requerit o la sensibilitat a l'ample de banda de la xarxa amb la que cada *Computing Element* interactua amb la resta de components.

A més d'estes característiques pròpies dels recursos que són capaços d'executar treballs seqüencials i paral·lels, cal tindre en compte una dada ja analitzada en apartats anteriors, i és la quantitat de recursos MPI disponibles en una infraestructura *Grid* com la d'EGEE. Com ja s'ha vist, percentualment són al voltant d'un 25% dels recursos seqüencials (per a treballs amb un sol node) i aquest percentatge va disminuint ràpidament si s'incrementa la demanda de nodes i si, a més, s'exigeix una configuració òptima del sistema. Açò pot representar una dada decisiva a l'hora de decantar-se per una estratègia o una altra, ja que la realitat actual de la infraestructura no permet fer desplegaments enfocats a experiments massius amb una estratègia paral·lela de memòria distribuïda amb MPI.

A les següents figures hi ha quatre comparatives diferents, basant-se en el temps invertit pels dos enfocaments, la memòria principal requerida, la memòria secundària necessària i els bytes totals descarregats en els dos sistemes. A la Figura 20 hi ha una comparativa dels temps globals dels dos processos, on es veu clarament la superioritat de la versió seqüencial. No obstant, se poden analitzar més aspectes que poden ser claus a l'hora de triar una estratègia i una altra, ja que és possible que la versió seqüencial siga més ràpida però no sempre puga fer front als requeriments de memòria principal, secundària o de bytes descarregats.

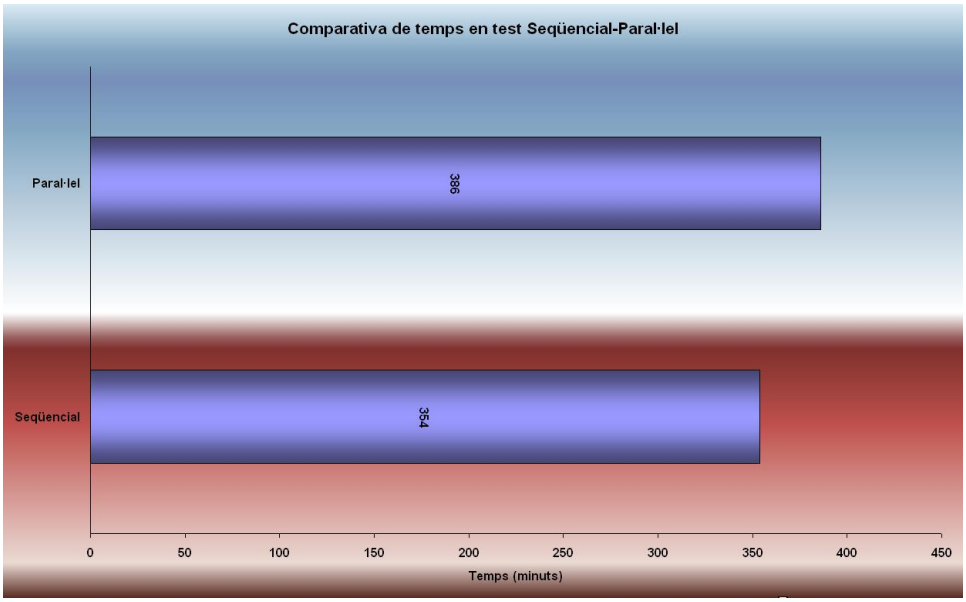

Figura 20: Comparativa de temps en el test seqüencial-paral·lel.

A la Figura 21 es pot apreciar la segona gràfica comparativa entre les dues versions estudiades. En aquest cas, la comparació és sobre la memòria principal requerida. La gràfica no deixa lloc a dubtes i els requeriments de la versió seqüencial, tot i ser assumibles, són més de quinze vegades superior a la versió paral·lela.

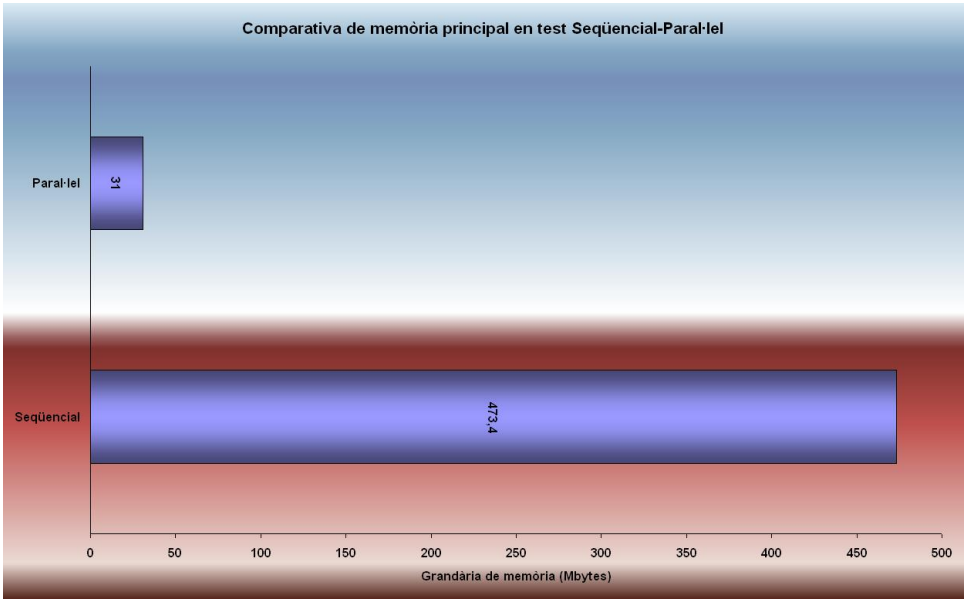

Figura 21: Comparativa de memòria principal en el test seqüencial-paral·lel.

La tercera gràfica es pot vore a la Figura 22, on hi ha una comparativa de la memòria secundària requerida en les dues versions. Novament, les necessitats de la versió seqüencial són 20 vegades superiors a les de la versió paral·lela.

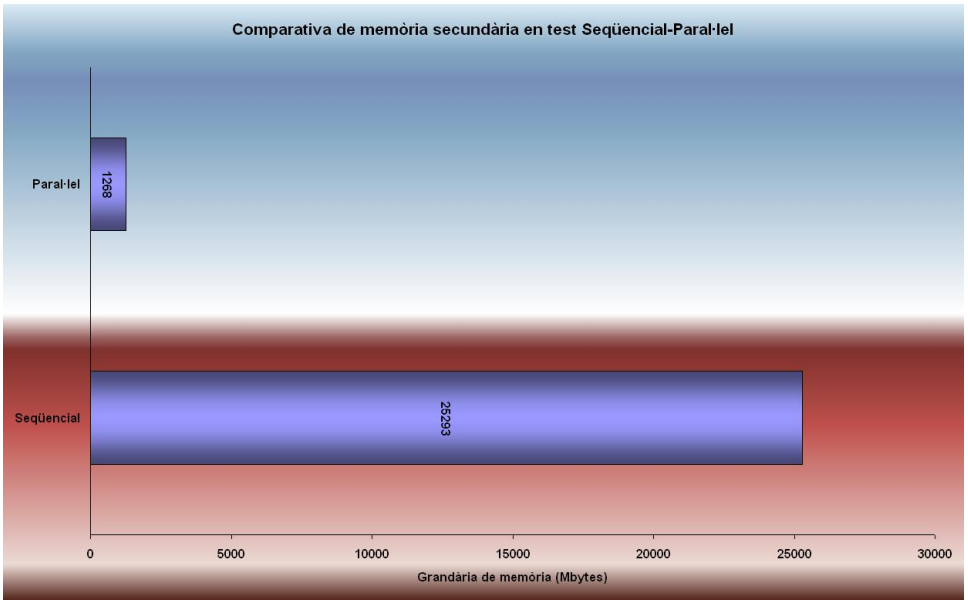

Figura 22: Comparativa de memòria secundària en el test seqüencial-paral·lel.

La darrera gràfica està disponible a la Figura 23 i compara els bytes que cal descarregar en les dues versions. La diferència entre les versions és també molt important, ja que en la versió seqüencial s'han de descarregar necessàriament 19 vegades més bytes que en la versió paral·lela.

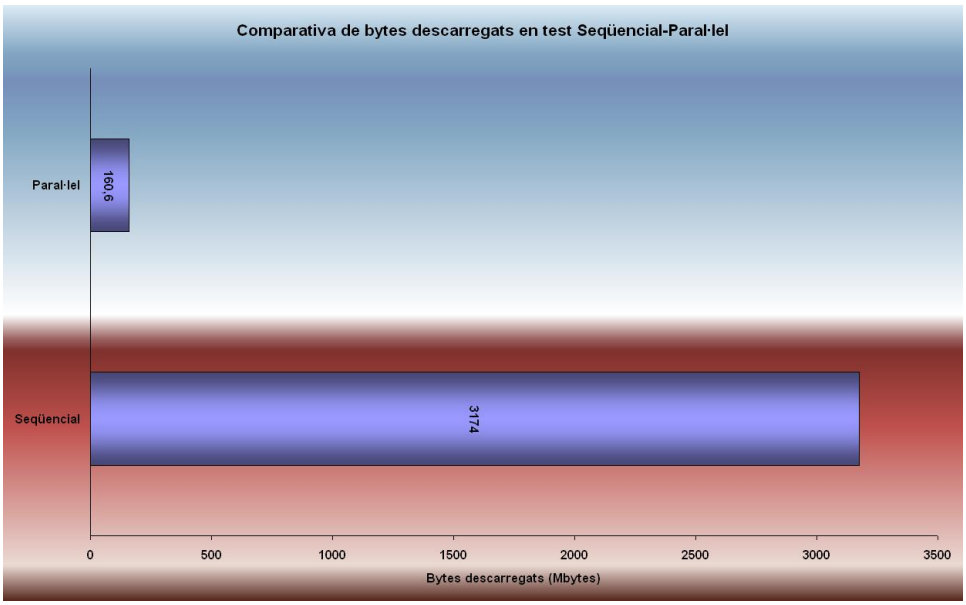

Figura 23: Comparativa de bytes descarregats en el test seqüencial-paral·lel.

Analitzats tots els aspectes, sembla necessari reconsiderar la forma en que es valora el cost temporal a l'hora de triar entre dues versions de d'una mateixa utilitat. Mentre que la diferència de temps és molt menuda (menys d'un 9%), la diferència en la resta d'aspectes sol ser de vint a un, cosa que pot provocar que la majoria dels recursos no estiguen en disposició de fer front a eixa demanda.

A sobre, cal remarcar que hi haurà cops en què aquest tipus de comparativa no serà possible ja que, com s'ha vist al capítol sobre la integració d'aplicacions bioinformàtiques a una infraestructura *Grid*, no sempre serà possible particionar un treball en sub-treballs i, per

tant, una versió MPI serà l'única forma d'accelerar el temps d'un experiment (contràriament al que passa en el cas BLAST).

Finalment, els resultats oferts per aquest experiment encoratgen a treballar per tal de tenir més recursos MPI disponibles dintre de la infraestructura *Grid* emprada, en aquest cas, la d'EGEE. Incrementant el nombre de recursos i millorant la seua configuració, per exemple, instal·lant versions MPI que traguen el màxim profit de la xarxa de comunicació pròpia del recurs. D'aquesta manera, és previsible una millora substancial per a aquelles aplicacions que només poden accelerar-se mitjançant una versió paral·lela i s'evitarà en gran mesura la rèplica innecessària d'espai en memòria principal i secundària, a més dels bytes que cal descarregar per a la seua correcta execució.

5.3. Paral·lelisme de tres capes

Als darrers tests, s'han emprat aplicacions d'ús corrent en l'àmbit de la biomedicina per a fer certes comparatives i traure conclusions sobre la forma d'abordar problemes on l'executable a gastar és seqüencial i altres on l'executable és paral·lel. En el present apartat, es pretén integrar al paral·lelisme *Grid* i el paral·lelisme de memòria distribuïda un nivell més de paral·lelisme en forma de paral·lelisme de memòria compartida. Per a tal efecte es gastaran els POSIX Threads (presentats a l'apartat corresponent de l'estat de l'art) i es realitzaran diverses implementacions d'un mateix algorisme (el dels K veïns més propers) per tal de fer una comparativa de temps segons la versió utilitzada.

S'ha seleccionat el mètode dels K veïns més pròxims per què es tracta d'un mètode d'implementació molt senzilla, ja que l'objectiu no està en el desenvolupament dels executables sinó en traure conclusions de com pot millorar les prestacions una combinació de les tres tecnologies paral·leles gastades. El treball es concentrarà en millorar les prestacions generals del sistema. L'anàlisi de la precisió i bonança de les prediccions (la qual cosa és depenent del mètode gastat i no de la implementació del mètode) no estaran en el nostre marc d'estudi.

Un cop ja es coneix el problema específic sobre el qual es treballa (el mètode KNN), es proposarà una arquitectura per a la aplicació (l'arquitectura del paral·lelisme de tres capes) gastant les tres tecnologies comentades. A partir d'ahí, es farà una revisió de la implementació, començant amb breus comentaris sobre les tres tecnologies emprades i la interfície elegida. Finalment es farà una ullada als resultats obtinguts amb la solució proposada i s'establiran un conjunt de conclusions sobre el treball realitzat en aquesta secció.

5.3.1. Arquitectura

La decisió de gastar tres nivells de paral·lelisme rau en la dificultat de controlar una gran quantitat de dades (especialment amb bases de dades epidemiològiques que contenen diversos milions de registres) i els pobres

resultats obtinguts amb les aproximacions seqüencials tradicionals. El propòsit està basat en realitzar un conjunt d'experiments amb diferents valors de K per al mètode K-NN per tal de determinar el valor òptim d'acord amb el valor de K que minimitza l'error en un test de validació creuada. Un cop es té el valor òptim per a K, es podrà procedir a classificar els registres no etiquetats amb el mètode K-NN.

El principal objectiu del treball serà reduir el temps de còmput especialment per a l'anàlisi i el suport epidemiològic. L'avaluació de l'error per a cada valor de K necessita unes 18 hores de CPU gastant un computador actual i una base de dades d'un milió de registres amb 20 camps per registre. Per tant, una procés complet d'optimització de 10 valors diferents de K tardaria més de 7 dies de CPU, la qual cosa converteix al problema en difícilment tractable en un entorn de producció.

A pesar de tot, el procés és intrínsecament paral·lel, presentant dos clars nivells de paral·lelisme. En un primer nivell, cada avaluació de l'error gastant diferents valors de K és totalment independent, ja que consisteix en la computació del mateix procés de classificació i de validació creuada per a cada valor de K, gastant la mateixa base de dades. Aquest procés consisteix en calcular les K distàncies mínimes a tots els registres i seleccionar un bloc com al conjunt de test. Aquest bloc serà reetiquetat gastant la resta de base de dades com a conjunt d'entrenament. Les etiquetes són assignades considerant les etiquetes dels K veïns més pròxims. Aquest procés serà repetit gastant diferents blocs de la base de dades com a conjunts d'entrenament per tal de cobrir la totalitat de la base de dades. Cada validació serà independent de la resta de validacions amb blocs diferents. A pesar d'això, el cost computacional d'aquest procés és de l'ordre de cN^2, sent c el nombre de FLOPS necessaris per a calcular una distància entre dos registres i N el nombre de registres a la base de dades (que afecta directament al cost de comunicacions).

Per tant, una solució de compromís serà aplicada per tal d'obtindre les màximes prestacions considerant la millor granularitat. En el marc d'aquest escenari, tres aproximacions paral·leles poden ser considerades. Les condicions de cadascuna de les aproximacions pel que fa al problema tractat són:

- Computació Grid.

  Aquesta tècnica implica la granularitat més grossa. La computació *Grid* es basa en l'ús concurrent de diferents recursos de còmput en diferents dominis administratius en una aproximació similar a una cua de tipus *batch* de gran escala. La comunicació entre recursos no està usualment disponible per culpa de les grans latències, la configuració interna dels nodes en els proveïdors de recursos i el sobrecost de les polítiques de seguretat. L'accés a les dades per tant és majorment realitzat a través del procés se submissió i dels rebostos compartits, gastant protocols semblants a FTP. En aquest paradigma, l'entitat executable mínima és el treball, interactuant amb la resta de treballs mitjançant els fitxers d'entrada i eixida.

- Computació paral·lela per pas de missatges.

Aquesta tècnica està demostrat que és molt eficient en la majoria de problemes de gra mitjà en els quals els costos de comunicació són d'un ordre de magnitud inferior als costos de computació. Típicament, els treballs són pràcticament simètrics i s'executen en nodes homogenis connectats amb una xarxa molt ràpida. Les polítiques de seguretat no són aplicades en la comunicació i l'intercanvi de dades es realitza mitjançant el pas de missatges.

- Computació paral·lela per memòria compartida.

Aquesta tècnica comprén el gra més fi de paral·lelisme. És aplicable en entorns molt acoblats i homogenis. En el cas dels *POSIX Threads*, hi haurà diferents fils d'execució concurrents amb un programa comú però treballant amb diferents fragments de dades. Les dades s'intercanviaran mitjançant regions compartides i mecanismes de contenció. Generalment, el factor d'escalabilitat d'aquestos sistemes és baix, degut a les restriccions de maquinari, però les acceleracions obtingudes són bones.

El nostre propòsit és el de combinar aquests tres nivells per a aconseguir les màximes prestacions. Amb les aproximacions de memòria compartida, només s'obtindran acceleracions menudes (limitades pel nombre de processadors disponibles) i són necessaris supercomputadors de memòria compartida. La combinació de les aproximacions amb memòria distribuïda i memòria compartida incrementarà notablement l'acceleració, ja que les granges de computadors poden aconseguir sense pèrdues de prestacions diverses desenes de nodes bi-processadors. A pesar de tot, i considerant que el problema és massivament paral·lel, es poden gastar configuracions molt més potents d'una forma eficient. Per tant, l'ús coordinat de diverses granges de computació és una opció raonable considerant la disponibilitat d'aquestos sistemes. En aquest cas, la computació *Grid* constitueix una forma eficient d'organitzar i controlar diferents recursos de còmput en diferents dominis administratius. Aleshores, amb la finalitat d'aconseguir les màximes prestacions, s'ha decidit de combinar les tres tècniques de computació paral·lela: tecnologia *Grid*, programació amb MPI i els fils de POSIX.

Considerant les diferents característiques de cada aproximació, el problema de la classificació pot ser estructurat de manera que s'obtinga la màxima eficiència per a cadascuna de les aproximacions. D'acord amb això, s'ha triat la infraestructura EGEE amb el *middleware* actual LCG-2.7. Es gastarà una interfície per línia de comandaments (CLI) per tal d'implementar els *scripts* i els programes per al llançament automàtic de diversos experiments per a diferents valors de K. Cada experiment es realitzarà concurrentment per diversos processos MPI amb la particularitat que el conjunt de mostres d'entrenament serà dividit entre els processos MPI existents. Finalment, es dividirà el conjunt de test en diversos subblocs per a computar-se amb fils de POSIX creats dins del procés MPI i executats pels distints processadors del node. Un exemple de diagrama en arbre d'aquesta aproximació és la impressió de la Figura 24**¡Error! No se encuentra el origen de la referencia.**. En aquesta figura es pot vore l'evolució del treball. Comença a l'arrel de l'arbre amb la submissió de K

treballs *Grid* LCG diferents (on el valor K determinarà per a cada treball LCG el nombre de veïns per als quals són calculades les distàncies). En una segona fase, els processos paral·lels MPI són executats. Hi haurà el mateix nombre de processos MPI que blocs de validació creuada, encara que altres factors del gra de paral·lelisme poden ser analitzades en el futur. En la tercera fase, els processos MPI són desplegats en forma de fils d'execució, d'acord amb les característiques dels recursos de maquinari a emprar i dels resultats experimentals.

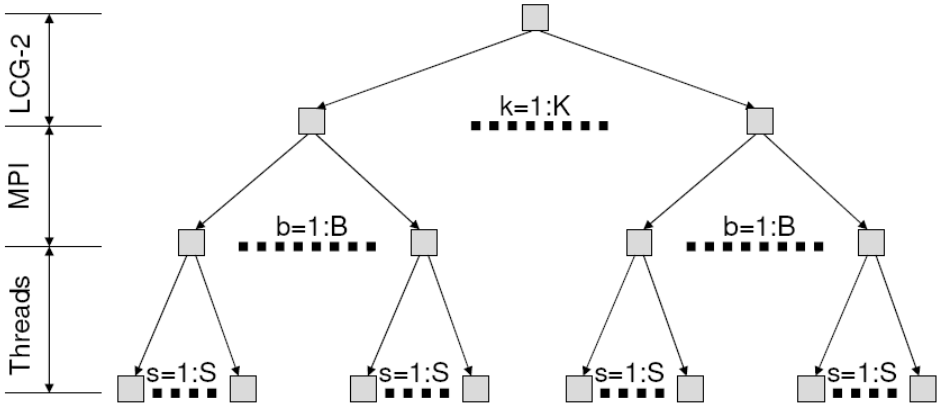

Figura 24: Esquema del paral·lelisme de tres capes

5.3.2. Implementació

La implementació del model de tres capes principalment considera tres components:

- El mòdul K-NN paral·lel. Aquest component implementa l'algorisme de classificació i validació creuada gastant MPI i fils de POSIX. Es tracta d'un executable autònom que pren com a entrada la referència al fitxer dels registres etiquetats, la referència als fitxers que es pretén etiquetar i el valor de K que es gastarà. El programa produeix com a eixida un fitxer diferent depenent del que es demane, sent possible mostrar les etiquetes assignades als registres objectiu o obtenir un fitxer amb resums estadístics.
- *Scripts* per a *Grid*. S'implementa la forma de seleccionar els recursos de còmput més adients, els fitxers de descripció de treball, el script de posada en marxa per a l'executable paral·lel, la submissió dels treballs i el seu monitoratge i obtenció de resultats. Totes aquestes tasques estan implementades a través de *scripts* que fan ús del CLI.
- Interfície *Java*. S'implementa una interfície amigable per tal de seleccionar les dades i els paràmetres per als treballs *Grid* i per a obtindre els resultats.
- El mòdul K-NN paral·lel comprén els nivells de computació amb MPI i amb fils de POSIX. L'execució sincronitzada amb diferents instàncies d'aquest mòdul de processament es realitza a través de scripts Grid. La Figura 25 mostra el procés global, d'acord amb un

punt de vista cronològic. En la part superior del diagrama es poden vore els subprocessos classificats per funcionalitat i en la part baixa estan classificats pel tipus de tecnologia emprada.

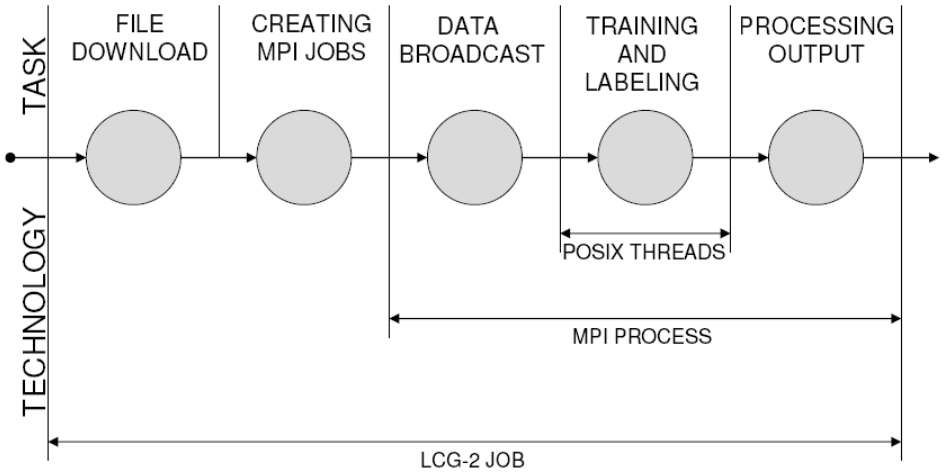

Figura 25: Esquema del procés global

5.3.2.1. Nivell de computació Grid

Una aplicació típica de la tecnologia *Grid* són les execucions multiparamètriques. La dificultat en establir comunicacions eficients entre treballs sotmesos de forma independent en un entorn *Grid* ha suposat tradicionalment un gran entrebanc. En el cas que es planteja, diferents experiments amb diferents valors del paràmetre K en el mètode dels K-NN, constitueix de forma clara una tasca multiparamètrica, la qual pot ser realitzada amb diferents treballs *Grid* de LCG. La infraestructura *Grid* seleccionada és, per tant, EGEE (*Enabling Grids in E-sciencE*). Aquesta és la infraestructura en producció més gran disponible per a la recerca en el món sencer, integrant a finals del 2006 a més de 29.000 computadores i més de 40 Petabytes de capacitat d'emmagatzemament al llarg de 177 *sites*. Aquesta infraestructura executa treballa actualment amb *gLite* 3.0. *gLite* 3.0 i LCG 2.7 comparteixen la major part dels seus components, cosa que fa la migració d'un a l'altre molt senzilla. Tant LCG 2.7 com *gLite* 3.0 estan són *middlewares Grid* orientats a *Batch* i tenen la mateixa estructura computacional, la qual comprén els següents components:

Recursos de còmput. Els recursos de còmput estan organitzats en *Computing Elements* (CEs) i *Worker Nodes* (WNs). Els CEs són els *front-ends* i punts d'entrada visibles a les granges computacionals formades per WNs. Els CEs implementen les cues batch necessàries per a controlar els treballs dintre dels WNs i vigilar l'estat tant dels treballs com dels recursos.

- Recursos d'emmagatzemament. Els fitxers a la infraestructura EGEE s'emmagatzemen de forma distribuïda en diversos *Storage Elements* (SEs).
- Control de càrrega. El destí d'un treball (un cua en un CE) pot ser directament seleccionat per l'usuari, encara que la forma més eficient és confiar en el sistema de control de càrrega (WMS).

- Catàleg d'emmagatzemament. Les dades emmagatzemades als SEs estan organitzades en catàlegs d'emmagatzemament. Els catàlegs d'emmagatzemament s'encarreguen de mantenir un registre dels fitxers emmagatzemats.
- Control d'usuaris. Els usuaris estan agrupats en organitzacions virtuals (VOs). Típicament, els usuaris d'una VO tenen els mateixos nivells d'autorització per a accedir als recursos. Açò redueix el cost de mantindre un control individual en les polítiques de seguretat.
- Sistema d'informació. La informació del sistema (estat dels treballs i dels recursos, principalment) és publicada en un model jeràrquic pels diferents recursos i el sistema de monitorització.

Per tal d'executar un treball en l'entorn LCG cal escriure un fitxer de descripció de treball, d'acord amb l'especificació del *Job Description Language* (JDL). Aquest fitxer defineix els arxius d'entrada i eixida, l'executable, els paràmetres de l'execució i els requeriments del programa. L'executable és típicament un *shell-script* que copia totes les dades necessàries localment al recurs i realitza altres passos preliminars (com ara re-compilació, canvi de permisos d'execució, etc.). El treball és sotmés a través de comandaments específics o de crides a una API i s'entra en un cicle d'estats (sotmés – esperant – preparat – programat – executant-se – finalitzat – eixida disponible – netejat). Un cop el treball LCG ha estat assignat a un *Computing Element*, un *shell-script* s'executa. Les instruccions inicials del *shell-script* s'encarreguen de portar les bases de dades guardades al *Grid*, incloent tant els registres d'entrenament com els de test, ja que no només es realitzarà l'entrenament sinó també es realitza la classificació en la darrera fase del treball. Aquesta aproximació reduirà el temps d'espera en el *Grid*. Quan totes les dades necessàries hagen estat descarregades en els nodes de còmput, el procés de classificació ja pot començar. Aquest procés es realitza gastant un procés MPI que serà el responsable de fer un test amb un bloc assignat. El sumatori de tots els errors dels test de bloc serà l'error de la validació creuada, és a dir, la informació que justament estem buscant per tal de decidir quin és el valor òptim per a la K i així gastar aqueix valor per a classificar els registres no etiquetats.

Les tasques que han de ser implementades per a aconseguir la funcionalitat descrita són:

- Selecció del recurs de còmput adient. Els recursos en el *Grid* d'EGEE són accedits a través del WMS. Aquestos recursos són seleccionats d'acord amb les característiques del treball i les polítiques de la VO. Els recursos són ordenats d'acord amb un barem configurable, on normalment es té en compte la proximitat als recursos d'emmagatzematge on estan les repliques de les bases de dades gastades i altres criteris per a millorar les prestacions (principalment el nombre de CPUs lliures o la longitud mitjana històrica de la cua de submissió).

- Submissió i resubmissió. Un cop els recursos estan identificats, les dades d'entrada per a cada treball s'empaqueten i els treballs són sotmesos amb tota la informació requerida (incloent referències a les bases de dades guardades). L'estat del treball és periòdicament monitoritzat i els treballs són resotmesos a un nou recurs de còmput en cas que s'haja excedit un temps d'espera predeterminat en el recurs inicialment assignat.
- Monitoratge i obtenció de resultats.. Els treballs que s'estan executant es monitoritzen amb els *scripts* corresponents. Un cop finalitzats, les dades obtingudes són descarregades I l'usuari rebrà una notificació via *e-mail* (obtés automàticament del *Distinguished Name* del certificat o passat com a paràmetre).
- El resultat final de tot el procés és l'error de la validació creuada per a una execució de K-NN amb un valor específic de K. Els resultats són presentats tan prompte estan disponibles i ordenats per la magnitud de l'error.

5.3.2.2. Nivell de computació MPI i fils de POSIX

L'executable MPI és un programa autònom que calcularà l'avaluació de la distància, la validació creuada i l'etiquetatge dels registres d'una base de dades.

El procés 0 de MPI serà el responsable de carregar i distribuir les bases de dades a la resta de processos MPI que s'executen en altres nodes del *cluster* seleccionat en la infraestructura *Grid* per a cada treball. Les bases de dades estan replicades en tots els processadors. El càlcul de la distància requereix tenir disponibles tots els registres etiquetats de la base de dades cada cop. Altres distribucions podrien ésser considerades si la memòria fos insuficient, encara que necessitarien d'un cost addicional en quant a comunicacions, ja que seria necessari un intercanvi de dades entre processos.

La distribució de grans quantitats de dades no implica una important penalització ja que la comunicació es realitzaria dins del propi *cluster* i no a través del *Grid*. El procés MPI 0 a més s'encarrega de normalitzar la base de dades per tal d'assegurar-se que tots els camps de cada registre són considerats amb el mateix pes. Els registres es distribueixen uniformement entre els nodes i dins de cada node uniformement entre els fils. Una llista dels K veïns més pròxims es va actualitzant al llarg del procés. Aquest procés s'executa per al bloc de registres seleccionat com a conjunt de test en cada computador.

Els errors són calculats com el nombre d'etiquetes assignades erròniament. Finalment, les etiquetes són assignades al final del procés per tal de reduir el sobrecost que suposaria redistribuir les dades una altra vegada i haver d'entrar de nou als sistemes de cues. Cada bloc de registres del conjunt de test és processat per un conjunt diferent de fils d'execució. Els fils de POSIX són creats en cada procés MPI i s'encarreguen de calcular la distància i etiquetar una porció del conjunt de test. L'ús de fils de POSIX

permet traure més partit de l'eficiència de la capacitat dels multiprocessadors que actualment tenen els *clusters*. A més, el procés no comporta conflictes ni en l'accés d'escriptura a variables comunes ni a la sincronització.

Els experiments també demostren que les capacitats del *hyper-threading* proveeixen un factor de guany addicional en l'acceleració, sent capaç de tindre més fils d'execució dels processadors disponibles. El cost moderat d'aquest procés (d'ordre quadràtic) i la gran quantitat de dades que ha de ser intercanviada fa aquest problema més propici per a la computació paral·lela que no per a la computació *Grid*. La complexitat del problema podria augmentar si puja el cost del càlcul de les distàncies (ara mateix es gasta una distància euclidiana homogènia) amb diferents pesos o distàncies mètriques per als diferents camps, o considerant errors mètrics més complexos, com ara les distàncies a l'etiqueta correcta.

El desenvolupament d'aplicacions MPI en diferents nodes *Grid* és un problema que actualment aborden els projectes Grid-MPI. Grid-MPI és una aproximació *Grid* al *Message Passing Interface*. Encara que Grid-MPI posa en marxa el desenvolupament d'aplicacions paral·leles MPI a gran escala, les infraestructures actuals no el soporten. La principal raó és que els WNs normalment poden comunicar-se dins del *cluster* i, encara que les comunicacions d'entrada estan permeses, la comunicació directa entre WNs de diferents recursos no és possible encara. Des del punt de vista de l'usuari, el recurs és el CE. A més, els recursos distribuïts geogràficament pel *Grid* es vorien afectats per la ineficiència de les comunicacions per a aplicacions de gra fi, per la qual cosa el paral·lelisme de gra gros és clarament el més adient per a l'estat actual dels sistemes *Grid*.

5.3.2.3. Interfície d'usuari

Per tal de facilitar el procés de creació de nous experiments, llançar treballs i monitoritzar els resultats, s'ha implementat una interfície gràfica basada en *java*. Aquesta interfície executa coordinadament els *scripts* necessaris per a fer la submissió dels treballs *Grid*, la seua monitorització i la recepció de resultats. La interfície fa possible que un usuari autenticat i autoritzat puga entrar al sistema i fins i tot pujar els conjunts de test i d'entrenament a un SE específic. Els CEs on es llançarà també seran seleccionats d'acord amb la disponibilitat de recursos de còmput, el suport de MPI i la proximitat al SE que publica una rèplica a les bases de dades emprades. Aleshores, els treballs són automàticament construïts considerant el rang de valors de K que s'ha especificat.

Els treballs se sotmeten a través de la interfície i el seu estat és monitoritzat tant a nivell global (percentatge de treballs en cada estat) o de forma individualitzada per treballs. També hi haurà previst un sistema per a resotmetre treballs en cas de problemes amb el CE escollit (encara que s'haja consumit el nombre de reintents especificats al JDL) o també en casos en què el treball està en espera o en cua massa temps, podent triar en aquestos casos un recurs de còmput diferent. Finalment, el resultat dels treballs serà consultat dinàmicament. L'usuari serà notificat tan prompte

com el treball haja finalitzat via *e-mail*. La Figura 26 mostra un parell d'instantànies de la interfície de l'aplicació.

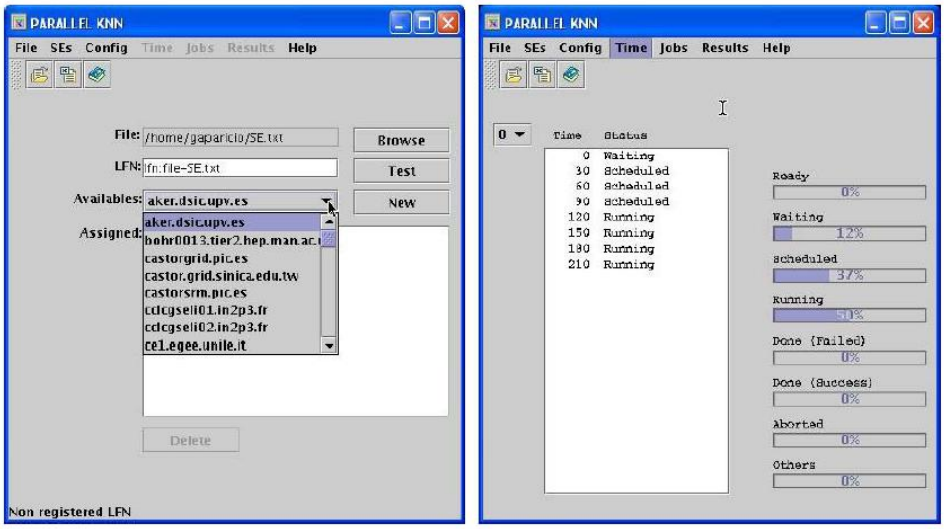

Figura 26: Dues instantànies de la interfície d'usuari de l'aplicació

5.3.3. Resultats

S'han fet diversos experiments en quatre escenaris. El primer, gastant l'aplicació WEKA per línia de comandaments, específicament la classe *java* K-NN (la versió 3.4.5 de l'amplament estesa i gastada ferramenta de classificació); segon, gastant una ferramenta seqüencial implementada en "C"; tercer, gastant una ferramenta paral·lela amb MPI i, quart, gastant la ferramenta K-NN amb tres capes de paral·lelisme.

S'han gastat dues bases de dades d'entrenament diferents, una amb 100.000 registres etiquetats i una altra amb 1.000.000 de registres etiquetats. El nombre de camps de les dues bases de dades era de 20, a més del camp d'etiqueta. Les prestacions de la versió implementada en llenguatge *C* era molt més eficient que WEKA, en gran part degut al fet que WEKA està implementada en *Java*. S'observa que apareix una acceleració lineal gastant la tecnologia *Grid*, ja que els experiments són independents.

El guany en l'aproximació paral·lela MPI ha estat sobre un factor 9.5 amb 10 biprocessadors (PIII Xeon 3GHz en una xarxa de torus SCI 3D) i el guany gastant 4 fils en cada node està al voltant d'un factor addicional de 1.5. Es van seleccionar 4 fils considerant que cada node té dos processadors amb *hyper-threading* i és així com més altes prestacions s'obtenien per al citat programa. El guany no és tan lineal com en altres casos ja que no tot el procés s'ha pogut implementar amb fils, lògicament. Per tant, el guany obtingut amb MPI i fils està sobre 15, i l'avantatge sobre WEKA s'ha de multiplicar encara per 6, o siga, 90. A més, cal afegir el que ja s'ha mencionat abans, i és que el *Grid* escala linealment. A la Figura 27 hi ha un resum dels resultats obtinguts. Aquesta figura reflecteix l'acceleració comparant el procés seqüencial amb WEKA, l'aplicació seqüencial en *C*, l'aplicació paral·lela en MPI i el definitiu procés amb paral·lelisme de tres capes.

Figura 27: Resultats d'acceleració

Després de la revisió dels resultats, es pot concloure que el resultat del treball ha estat molt encoratjador. L'aproximació basada en el paral·lelisme de tres capes és una forma molt efectiva d'aconseguir les millors prestacions i oferir una visió optimista de la mineria de dades al *Grid*.

La classificació dels registres, incloent la identificació del valor òptim de K en el mètode K-NN en una base de dades d'un milió de registres, tardaria menys de sis hores, és a dir, un simple execució que es podria fer en una sola nit. Comparant aquestos resultats, es demostra que s'està davant d'una acceleració major de 90 respecte al seu equivalent seqüencial WEKA, cosa que suposaria més d'un mes de temps d'espera. A tot açò caldria afegir el fet que només estem parlant d'un recurs *Grid* i aquesta taxa d'acceleració es podria vore multiplicada pel nombre de recursos emprats. O siga, si es gastaren 10 CEs similars a l'emprat, l'acceleració ja seria de 900.

Si es té en compte que els esforços han estat dirigits a una predicció biomèdica, aquest avantatge farà possible, per exemple, la classificació d'informació registrada en l'atenció primària cada dia (de l'ordre de milions de registres) de forma automàtica i ràpida.

6. CONCLUSIONS

Finalment arriba el moment de fer un repàs de les principals conclusions del present treball. Per això s'ha dividit el capítol de conclusions en un sub-capítol amb les contribucions pròpies en aquest treball, després un altre sub-capítol amb el conjunt de publicacions generades a partir del treball desenvolupat a la present tesi de màster i, per últim, un repàs als treballs futurs que queden pendents i s'hauran de resoldre pròximament.

6.1. Contribucions pròpies.

L'objectiu de la present tesi, com el seu títol indica, era la d'integrar diferents tecnologies paral·leles per tal de fer front als requeriments d'un grup d'aplicacions bioinformàtiques. Per això, s'ha fet una anàlisi tant de les tecnologies com de les ferramentes disponibles per tal de fer una fotografia, més o menys aproximada, de la realitat tecnològica. Relacionat amb açò, s'ha fet incís en la forma en què una infraestructura *Grid* concreta, com la d'EGEE, ha estat capaç d'integrar totes aquestes tecnologies i l'oferta real actual.

També ha calgut analitzar un subconjunt d'aplicacions per a l'anàlisi genòmic, revisant les diferents versions ofertades i fent una caracterització inicial de la seua capacitat d'adaptació a entorns *Grid*.

Després de la tasca de documentació i anàlisi ha estat necessària una implicació en forma d'implementacions per tal de tractar de mesurar els problemes i avantatges sobre una infraestructura *Grid* real. Per això, en primer lloc s'ha desenvolupat un model genèric de desplegament per a experiments massius a la infraestructura d'EGEE, com bé podrien ser qualsevol dels estudis posteriors que s'han realitzat. En aquesta part s'han descrit els diferents components que conformen el sistema i dels elements de configuració que tindran un impacte directe en les prestacions dels experiments finals.

Finalment, s'ha realitzat una sèrie d'estudis, a través de diferents casos d'estudi, on s'ha pretés mesurar de forma empírica les possibilitats i característiques del sistema desenvolupat i avaluar l'impacte concret de diversos paràmetres claus del sistema, com podrien ser la distribució de la càrrega, el particionament de les dades o l'estratègia de paral·lelisme emprada, optant per versions seqüencials coordinades amb la interfície d'usuari de *gLite 3.0* o per versions paral·leles MPI, deixant la coordinació dels diferents processos en mans del propi *middleware* de pas de missatges. En concret, amb el treball desenvolupat s'han processat en poc més de 2 mesos de temps real l'equivalent a 10 anys CPU.

Per últim, a partir d'un algorisme senzill de classificació en el marc de la mineria de dades, s'han desenvolupat una sèrie d'aplicacions, integrant diferents tecnologies paral·leles com són la computació paral·lela de memòria compartida, de memòria distribuïda i *Grid*. I creant diverses versions d'una mateixa aplicació base s'ha aconseguit demostrar, a partir

de la comparació entre aquestes versions, que la combinació de les tres tecnologies no només ha estat exitosa sinó que és possiblement la solució lògica per a un progrés en la carrera per l'augment de les prestacions computacionals i la millor manera d'abordar problemes de gran magnitud.

En definitiva, la present memòria ha estat un viatge intens per la realitat de les tecnologies paral·leles en el marc d'una infraestructura *Grid*. Al llarg del treball s'han trobat moltes dificultats, normalment causades per la manca de correlació entre les promeses de la teòria i la contundència de la realitat pràctica. Entretant, s'han trobat motius d'esperança, espais fins ara inexplorats que cal explotar i raons per a la reflexió. La principal reflexió, en aquest sentit, segurament seria si està justificada la creixent obsessió pel maquinari quan la manca d'esforços en l'optimització i configuració del programari deixa molt que desitjar, i la seua capacitat de millora és molt superior a les prestacions promeses pel nou maquinari.

6.2. Publicacions desenvolupades.

En aquest apartat s'enumeraran un recull de publicacions generades al voltant del treball d'aquesta tesi de màster. La majoria d'elles són articles en congressos, mentre que hi ha també alguna contribució en *workshops* i un pòster en unes jornades.

[1] G. Aparicio and R. Isea. Estat de desenvolupament de les aplicacions biomèdiques aprovades en EELA. First Latin American EELA Workshop, Mérida (Venezuela), 24-26 d'abril de 2006.

[2] G. Aparicio, I. Blanquer, V. Hernández, et al. Blast2GO goes Grid: Developing a Grid-Enabled Prototype for Functional Genomics Analysis, "HealthGrids: Challenges and Opportunities", HealthGrid 2006 Conference Proceedings, IOS Press Studies in Health Technologies and Informatics (ISBN I-58603-617-3), Vol .120, Pag.: 194-204, 2006.

[3] G. Aparicio, I. Blanquer and V. Hernández. A Parallel Implementation of the K Nearest Neighbours Classifier in Three Levels: Threads, MPI Processes and the Grid. VECPAR 2006 a Rio de Janeiro (Brasil), 10-13 de juliol de 2006. (ISBN 978-3-540-71350-0) Pag.:225-235.

[4] G. Aparicio, I. Blanquer, D. Segrelles and V. Hernández. BLAST in Grid (BiG): A Grid-Enabled Software Architecture and Implementation of Parallel and Sequential BLAST. Spanish Conference on Science Grid Computing, Madrid 1-2 de març de 2007. (ISBN 978-84-7834-544-1).

[5] R. Abarca, A. Acero, G. Andronico, G. Aparicio, C. Baeza, R. Barbera, I. Blanquer, M. Carrillo, D. Carvalho, J. Casado, J.L. Chaves, C. Cherubino, A. Cofío, J. Cruz, M. Diniz, M.T. Dova, I. Dutra, F. Echeverría, L. Enriquez, F. Fernández-Lima, F. Fernández-Nodarse, M. Fernández, V. Fernández, J.M. Gutiérrez, A. Hernández, V. Hernández, R. Isea, D. López, G. Lucet, B. Marechal, R. Mayo, R. Miguel, E. Montes, H.R. Mora, L. Nellen, L.N. Ciuffo, R. Pezoa, A. Porto, P. Rausch, A.J. Rubio-Montero, L. Salinas, E. Silva (2007) "Building a Network in Latin America: e-Infrastructure and Applications". Proceedings of the Spanish Conference on e-Science Grid Computing (ISBN 978-84-7834-544-1), pp. 83-96

[6] V. Hernández, I. Blanquer, G. Aparicio, Raúl Isea, J. L. Chaves, A. Hernández, H. R. Mora, D. López, M. Fernández, F. Blanco and R. Mayo (2007). "Biohealth in EELA". Proceedings of the NETTAB Conference. Vol. 7, pp. 145-156

[7] G. Aparicio, I. Blanquer and V. Hernández. Advances in the Biomedical Applications of the EELA Project. Geneva (Switzerland) HealthGrid 2007. Conference Proceedings, IOS Press Studies in Health Technologies and Informatics (ISBN 978-1-58603-738-3), Vol. 126, Pag.: 31-36

[8] EELA Application and tools customized. EELA applications. EU Deliverable D3.2.3, 25 de juliol de 2007. http://documents.eu-eela.org/record/787/files/EELA-D3.2.3-v1.12.pdf

[9] EELA Application impact report. All the EELA applications. EU Deliverable D3.3.3, 11 de gener de 2008. http://documents.eu-eela.org/record/939/files/EELA-D3.3.3-v1.15.pdf

[10] G. Aparicio, I. Blanquer and V. Hernández (2007). Metagenomic Analysis on the EELA Grid. Third EELA Conference Abstract book (ISBN)

[11] G. Aparicio, M. Pignatelli, I. Blanquer, A. Moya, V. Hernández and J. Tamames. Grid Computing for Metagenomics. Pòster a les VIII Jornades de Bioinformàtica 2007 a València.

[12] G. Aparicio, I. Blanquer and V. Hernández. A Highly Optimized Grid Deployment: the Metagenomic Analysis Example. Chicago (EUA) HealthGrid 2008. Conference Proceedings, IOS Press Studies in Health Technologies and Informatics. Pendent d'aprovació.

[13] BiG: BLAST in Grids. http://www.grycap.upv.es/bio

6.3. Treballs futurs.

Després del treball realitzat es pot concloure que hi ha moltes línies que s'han deixat pel camí i moltes altres que encara queden per cobrir. Els esforços destinats en el futur pròxim han de ser una combinació d'ambdues necessitats. Per una banda, cal publicitar els resultats obtesos en la combinació de tecnologies paral·leles per fomentar el seu ús i aconseguir una correcta configuració dels distints recursos de la infraestructura d'EGEE i així fer un considerable salt qualitatiu en les prestacions dels experiments realitzats. Per altra banda, cal continuar el treball en l'optimització tant de les característiques pròpies del desplegament com del sistema d'automatització plantejats a la memòria de la tesi.

Els models presentats caldrà analitzar si són exportables a altres tipus de tecnologies, com podria ser la tecnologia *Peer-to-Peer* (P2P), cosa que obriria el camp bioinformàtic a tot un seguit de nous reptes i possibilitats, dotant al sistema d'independència respecte a projectes com el d'EGEE amb un grau d'estabilitat impredible.

Finalment, és imprescindible treballar en el desenvolupament d'un conjunt de components de *middleware Grid* per a l'optimització de l'aprofitament dels recursos, fent especial èmfasi en els treballs *Grid* inherentment paral·lels, en una planificació eficient de fluxes de treballs, la millora de les interfícies i l'emmagatzemament segur de dades distribuïdes. També caldrà aprofundir en el disseny i implementació d'una sèrie de ferramentes, per al citat *middleware Grid*, que faciliten l'ús global de les infraestructures, com per exemple, la creació de federacions d'infraestructures *Grid*, la simplificació dels requeriments als nodes i la millora en el desplegament de les infraestructures.

REFERÈNCIES BIBLIOGRÀFIQUES

[1] Declaració de Bolonya. http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf

[2] I. Foster, C. Kesselman, S. Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3) (2001) http://www.globus.org/research/papers/anatomy.pdf

[3] Message Passing Interface Forum: MPI: A message-passing interface standard. (2003) http://www.mpi-forum.org/

[4] Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., Snir, M.: MPI: The Complete Reference, MIT Press, Cambridge, MA (1998)

[5] EGEE: Enabling Grids in e-Science. http://www.eu-eege.org/

[6] EELA: E-Infrastructure shared between Europe and Latin America. http://www.eu-eela.org/

[7] Drepper, U and I. Molnar: The Native POSIX Thread Library for Linux. (2003) http://people.redhat.com/drepper/nptl-design.pdf

[8] OpenMP application program interface, ver. 2.5, Technical report, http://www.openmp.org/, maig 2005.

[9] Introduction to Programming with Fortran with coverage of Fortran 90, 95, 2003 and 77: Ian Chivers and Jane Sleightholme. Springer Verlag, ISBN 1-84628-053-2, 2005.

[10] B. W. Kernighan, D. M. Ritchie, The C programming language, Prentice-Hall, Inc., Upper Saddle River, NJ, 1978

[11] Bjarne Stroustrup, The C++ Programming Language, Addison-Wesley Pub Co; Tercera edició (15 de febrer de 2000); ISBN 0201700735

[12] Proceedings Supercomputing '94, novembre 14-18, 1994, Washington, DC, EUA. IEEE Computer Society / ACM, 1994, ISBN 0-8186-6605-6

[13] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. S. (1994). PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press. http://www.csm.ornl.gov/pvm/pvm_home.html

[14] I. Foster, N. T. Karonis, A grid-enabled MPI: message passing in heterogeneous distributed computing systems (1998). Proceedings of the 1998 ACM/IEEE conference on Supercomputing, pàg. 1-11 (ISBN:0-89791-984-X)

[15] N. Karonis, B. Toonen, and I. Foster, MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, Journal of Parallel and Distributed Computing (JPDC), Vol. 63, No. 5, pàg. 551-563, maig 2003.

[16] PACX-MPI, The Grid-Computing library PACX-MPI Extending MPI for Computational Grids http://www.hlrs.de/organization/amt/projects/pacx-mpi/

[17] The Source for Java Technology. Sun Microsystems. http://java.sun.com

[18] I. Foster , C. Kesselman. The Grid: Blueprint for a new computing infrastructure (1998).

[19] LCG: World Wide Web Computing Grid. Distributed Production Environment of Physics Data Processing. http://lcg.web.cern.ch/LCG

[20] LightWeight Middleware for Grid Computing. http://glite.web.cern.ch/glite

[21] The Globus Project: A Status Report. I. Foster, C. Kesselman. Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop, pàg. 4-18, 1998. ftp://ftp.globus.org/pub/globus/papers/globus-hcw98.pdf

[22] Globus Toolkit Support for Distributed Data-Intensive Science. W. Allcock, A. Chervenak, I. Foster, L. Pearlman, V. Welch, M. Wilde. Proceedings of Computing in High Energy Physics (CHEP '01), setembre 2001. http://www.globus.org/research/papers/Globus.CHEP01.pdf

[23] "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". I. Foster, C. Kesselman, J. Nick, S. Tuecke, Open Grid Service Infrastructure WG, Global Grid Forum, juny 2002. http://www.globus.org/research/papers/ogsa.pdf

[24] Open Grid Services Infrastructure (OGSI) Version 1.0. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling; Global Grid Forum Draft Recommendation, juny 2003. http://www.globus.org/research/papers/Final-OGSI-Specification-V1.0.pdf

[25] The WS-Resource Framework. http://www.globus.org/wsrf

[26] GridFTP Protocol Specification. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, S. Tuecke. GGF GridFTP Working Group Document, setembre 2002. http://www.globus.org/research/papers/GridftpSpec02.doc

[27] GridFTP Update January 2002. W. Allcock, J. Bresnahan, I. Foster, L. Liming, J. Link, P. Plaszczac. Technical Report, gener 2002. http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf

[28] An Online Credential Repository for the Grid: MyProxy. J. Novotny, S. Tuecke, V. Welch. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, agost 2001. http://www.globus.org/research/papers/myproxy.pdf

[29] Grid Security Infrastructure. http://www.globus.org/security/overview.html

[30] P. Hallam-Baker, Eve Maler, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), OASIS Standard, novembre 2002. http://www.oasis-open.org.

[31] F. Pacini. Job Description Language How-to. http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.pdf

[32] T.M.Cover, P.E.Hart: Nearest neighbour pattern recognition. IEEE Trans. on Information Theory 13(1) (1967) 2127

[33] E. Frank, M. Hall, L.T.: Weka 3: Data Mining Software in Java. (2005) http://www.cs.waikato.ac.nz/ml/weka.

[34] Altschul,S.F., Gish,W., Miller,W., Meyers,E.W. and Lipman,D.J. (1990) Basic Local Alignment Search Tool. Journal of Molecular Biology 215, 403-410.

[35] National Centre for Biotechnology Information (NCBI) http://www.ncbi.nlm.nih.gov/

[36] mpiBLAST: Open-Source Parallel BLAST home page, http://mpiblast.lanl.gov/

[37] Conesa,A., Götz,S., García-Gómez,J.M., Terol,J., Talón,M., and Robles,M. (2005) Blast2GO: A Universal Tool for Annotation, Visualization and Analysis in Functional Genomics Research. Bioinformatics 21, 3674-3676.
[38] G. Aparicio, I. Blanquer, V. Hernández, et al. Blast2GO goes Grid: Developing a Grid-Enabled Prototype for Functional Genomics Analysis, "HealthGrids: Challenges and Opportunities", HealthGrid 2006 Conference Proceedings, IOS Press Studies in Health Technologies and Informatics (ISBN I-58603-617-3), Vol .120, pàg.: 194-204, 2006.
[39] A. Darling, L. Carey, and W. Feng. The Design, Implementation, and Evaluation of mpiBLAST**. 4**th International Conference on Linux Clusters: The HPC Revolution 2003 in conjunction with ClusterWorld Conference & Expo, juny 2003.http://www.mpiblast.org/downloads/pubs/cwce03.pdf
[40] Mr. Bayes 3: Bayesian phylogenetic inference under mixed models. Ronquist F., Huelsenbeck JP. http://mrbayes.csit.fsu.edu/
[41] Holder M. and P. O. Lewis. 2003. Phylogeny estimation: Traditional and Bayesian approaches. Nature Reviews Genetics 4: 275-284.
[42] Chenna R, Sugawara H, Koike T, Lopez R, Gibson TJ, Higgins DG, Thompson JD (2003). Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res, 31:3497-3500.
[43] ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Julie D. Thompson, Desmond G. Higgins and Toby J. Gibson. European Molecular Biolgoy Laboratory Postfach 102209, Meyerhofstrasse 1, D-69012 Heidelberg, Alemanya
[44] The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. Thompson JD, Gibson TJ, Plewniak F, Jeanmougin F, Higgins DG (1997). Nucleic Acids Research, 25:4876-4882.
[45] ClustalW-MPI: ClustalW Analysis Using Distributed and Parallel Computing, Kuo-Bin Li, Bioinformatics, 2003, 19(12), 1585--1586.
[46] Guia d'usuari de gLite 3.0 https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf
[47] MPI wiki page. http://www.grid.ie/mpi/wiki
[48] Grup de treball de MPI en EGEE-II. http://egee-intranet.web.cern.ch/egee-intranet/NA1/TCG/wgs/EGEE-II-MPI-WG-TEC-2.pdf
[49] Status of MPI in EGEE http://indico.cern.ch/materialDisplay.py?contribId=s3t2&sessionId=s3&materialId=slides&confId=a063547
[50] Interactive European Grid Project (int.eu.grid). http://www.interactive-grid.eu/
[51] The Sixth Framework Programme. http://cordis.europa.eu/fp6/
[52] The CrossGrid Project. http://www.crossgrid.org/
[53] Wide In Silico Docking On Malaria. . http://wisdom.healthgrid.org
[54] BioinfoGRID: Bioinformatics Grid Application for life science, http://www.bioinfogrid.eu/
[55] R. Casado et al. Large scale genome comparison in a GRID infrastructure. BioinfoGRID Symposium 2007 a Milà, Itàlia en desembre 2007
[56] P. Lio et al. Transcriptomics and phylogenetics applications in GRID. BioinfoGRID Symposium 2007 a Milà, Itàlia en desembre 2007

ANNEXE I: SHELL-SCRIPT DE TREBALL BiG SEQÜENCIAL

```
#!/bin/bash

CE_NAME=$1 ;
IDSESSION=$2 ;
EXP_ID=$3 ;
SEQ_FILE=$4 ;

mkdir proves ;
mv ${SEQ_FILE} proves/ ;
mv run_blast.sh proves/ ;
chmod +x proves/run_blast.sh ;
mv nr_access_index.pl proves/ ;
mv run_blast.pl proves/ ;
mv testolap.nr.pl proves/ ;
cd proves ;
PATH_LOCAL=`pwd`;

date ;
wget
ftp://ftp.ncbi.nih.gov/blast/executables/LATEST/ncbi.t
ar.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error baixant fitxer amb WGET" ;
fi ;

date ;
tar xzvf ${PATH_LOCAL}/ncbi.tar.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error descomprimint NCBI Toolkit" ;
fi ;

date ;
ncbi/make/makedis.csh ;
export PATH=${PATH_LOCAL}/ncbi/bin:$PATH ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error fent el make del NCBI Toolkit" ;
fi ;

date ;
mkdir db ;
lcg-cp --vo biomed
lfn:/grid/biomed/BiG/nr_notEukaryota.gz
file:${PATH_LOCAL}/db/nr_notEukaryota.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error baixant Base de Dades amb LCG-CP" ;
fi ;
```

```
date ;
gunzip ${PATH_LOCAL}/db/nr_notEukaryota.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error descomprimint la Base de Dades amb
GUNZIP" ;
fi ;

date ;

lcg-cp --vo biomed
lfn:/grid/biomed/BiG/nr_notEukaryota.index.gz
file:${PATH_LOCAL}/db/nr_notEukaryota.index.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error baixant Index de la Base de Dades amb
LCG-CP" ;
fi ;

date ;
gunzip ${PATH_LOCAL}/db/nr_notEukaryota.index.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error descomprimint Index de la Base de Dades
amb GUNZIP" ;
fi ;

date ;

${PATH_LOCAL}/ncbi/bin/formatdb -i
${PATH_LOCAL}/db/nr_notEukaryota -t nr_notEukaryota -p
T ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error formatant BD amb FORMATDB" ;
fi ;

date ;
OUTFILE=${EXP_ID}.${CE_NAME}.${IDSESSION}.txt ;

${PATH_LOCAL}/run_blast.sh ${PATH_LOCAL}/${SEQ_FILE}
blastx 1e-02 ${PATH_LOCAL}/db/nr_notEukaryota 5000
${PATH_LOCAL}/db/nr_notEukaryota.index ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error processant seqs amb BLASTALL" ;
fi ;

date ;
mv ${PATH_LOCAL}/${SEQ_FILE}.global.blast.out
${PATH_LOCAL}/${OUTFILE}
gzip ${PATH_LOCAL}/${OUTFILE} ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error comprimint eixida amb GZIP" ;
fi ;
```

```
date ;
lcg-cr --vo biomed -d ${VO_BIOMED_DEFAULT_SE} -l
lfn:/grid/biomed/BiG/FASTA/${OUTFILE}.gz
file:${PATH_LOCAL}/${OUTFILE}.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error pujant resultats amb LCG-CR" ;
fi ;

lcg-rep -d nephthys.dsic.upv.es --vo biomed
lfn:/grid/biomed/BiG/FASTA/${OUTFILE}.gz ;
error=$? ;
if [ ${error} -ne 0 ]; then
   echo "Error fent replica de resultats amb LCG-REP"
;
fi ;

date ;
cd .. ;
rm -rf ${PATH_LOCAL} ;
```