# Addressing Bandwidth Contention in SMT Multicores Through Scheduling

Josué Feliu, Julio Sahuquillo, Salvador Petit, and José Duato
Department of Computer Engineering
Universitat Politècnica de València
València, Spain
jofepre@gap.upv.es, {jsahuqui,spetit,jduato}@disca.upv.es

## ABSTRACT

To mitigate the impact of bandwidth contention, which in some processes can yield to performance degradations up to 40%, we devise a scheduling algorithm that tackles main memory and L1 bandwidth contention. Experimental evaluation on a real system shows that the proposal achieves an average speedup by 5% with respect to Linux.

## Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process management—*Scheduling*

## Keywords

bandwidth-aware scheduling; bandwidth contention

## 1. PROPOSED SCHEDULER

Algorithm 1 presents the pseudocode of the devised scheduler. It consists of process selection (lines 2-8) and process allocation (lines 9-12), which deal with main memory and L1 bandwidth contention, respectively, by balancing the memory requests over the workload execution time and the L1 requests among the L1 caches. Previously, the scheduler calculates the average main memory transaction rate of the workload following a similar approach to [2].

In the **process selection**, the proper set of processes is selected to be run during the following quantum. The process not executed for longer is always selected to avoid process starvation. Then, the remaining processes are selected using the fitness function, which quantifies the gap between the $TR_{MM}$ required by a given process and the average bandwidth remaining for each unallocated hardware thread [2].

In the **process allocation**, the selected processes are allocated to the cores. Since the experimental platform implements dual-threaded cores, the L1 bandwidth can be easily balanced by sorting the processes according to its $TR_{L1}$ and then, reiteratively, assigning the processes with highest and lowest bandwidth utilization to the same core [1].

## 2. EXPERIMENTAL EVALUATION

The experimental evaluation is carried out in an Intel Xeon E5645 processor, with six dual-thread SMT cores, a private L1 cache per core and a shared LLC. The algorithm has been implemented in a user-level scheduler. To evaluate the performance of the proposal, a set of ten 24-benchmark mixes was designed.

**Algorithm 1** Bandwidth-Aware Scheduler

**Require:** Prior calculation of the $AVG\_WK\_TR_{MM}$
1: **while** there are unfinished processes **do**
2: Gather $TR_{MM}$ and $TR_{L1}$ of the processes
3: $BW_{Remain} = AVG\_WK\_TR_{MM}$, $CPU_{Remain} = \#CPUs$
4: Select the process $p$ at the process queue head and update $BW_{Remain}$ and $CPU_{Remain}$
5: **while** # selected process < #CPUs **do**
6: Select the processes $p$ that maximizes:
$$\text{FITNESS(p)} = \frac{1}{\left|\frac{BW_{Remain}}{CPU_{Remain}} - TR^{p}_{MM}\right|}$$
7: Update $BW_{Remain}$ and $CPU_{Remain}$
8: **end while**
9: Sort the selected processes in ascending $TR_{L1}$
10: **while** there are unallocated processes **do**
11: Assign the processes $P_{head}$ and $P_{tail}$ with maximum and minimum bandwidth requirements to the same core
12: **end while**
13: **end while**

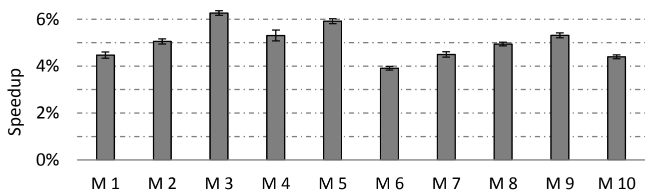


Figure 1: Speedup relative to Linux.

Figure 1 presents the speedup the devised scheduler achieves compared to the Linux scheduler across all the mixes using the average IPC with 95% confidence intervals. Results show that the scheduler effectively addresses bandwidth contention and improves the Linux performance by 5% on average.

## 3. ACKNOWLEDGMENTS

This work was supported by the Spanish Ministerio de Economía y Competitividad (MINECO) and Plan E funds, under Grant TIN2012-38341-C04-01, and by the Intel Early Career Faculty Honor Program Award.

## 4. REFERENCES

[1] J. Feliu, J. Sahuquillo, S. Petit, and J. Duato. L1-Bandwidth Aware Thread Allocation in Multicore SMT Processors. In *PACT*, pages 123–132, 2013.
[2] D. Xu, C. Wu, and P.-C. Yew. On mitigating memory bandwidth contention through bandwidth-aware scheduling. In *PACT*, pages 237–248, 2010.


*ICS'14,* June 10–13 2014, Munich, Germany.
ACM 978-1-4503-2642-1/14/06.
http://dx.doi.org/10.1145/2597652.2600109.