Thread and Memory Placement on NUMA Systems: Asymmetry Matters

Baptiste Lepers, Alexandra Fedorova (Simon Fraser University),
Vivien Quéma (Grenoble INP)

ATC 2015

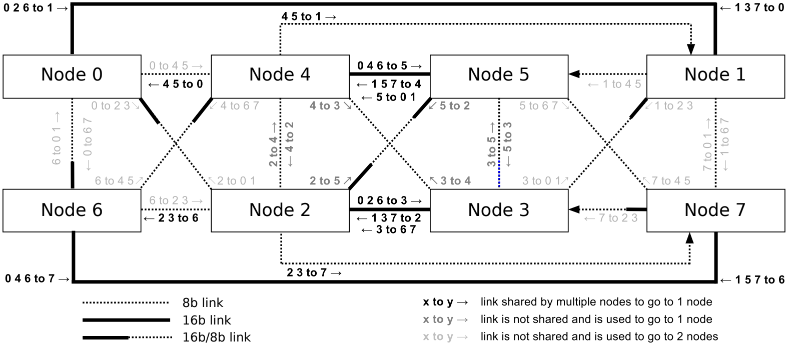
Introduction

Current threads and memory placement: minimizing hop-count (e.g. in Linux).

Contributions:

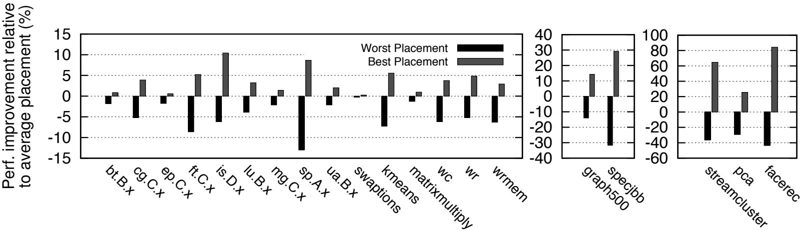
- Connections are asymmetric, bandwidth is more important than hops.
- *AsymSched* algorithm that dynamically places threads and memory.

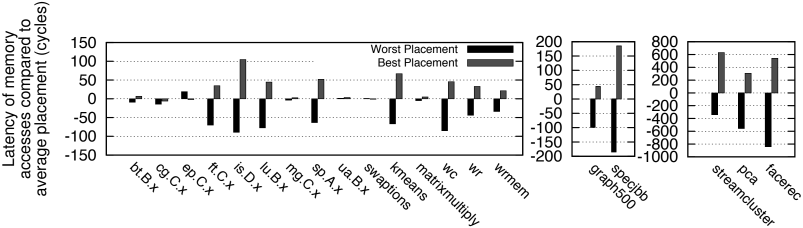
Inter-node bandwidths for 4 AMD Opteron 6272 processors



Measurements

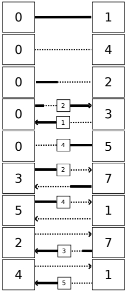
Applications running on 3 nodes, with different node placements.





More Measurements

streamcluster running on 2 nodes, with different node placements.



Master thread node	Execution Time (s)	Diff with 0-1 (%)	Latency of memory accesses (cycles) (compared to 0-1(%))	% accesses via 2-hop links	Bandwidth to the "master" node (MB/s)
-	148	0%	750	0	5598
-	228	56%	1169 (56%)	0	2999
0	228	56%	1179 (57%)	0	2973
2	168	15%	855 (14%)	0	4329
0	340	133%	1527 (104%)	98	1915
3	**185**	27%	1040 (39%)	98	3741
0	340	133%	1601 (113%)	98	1903
5	**228**	56%	1206 (61%)	98	2884
3	185	27%	1020 (36%)	0	3748
7	338	132%	1614 (115%)	98	1928
1	338	132%	1612 (115%)	98	1891
5	230	58%	1200 (60%)	0	2880
2	**167**	15%	867 (16%)	98	3748
7	225	54%	1220 (63%)	0	3014
4	230	58%	1205 (60%)	0	2959
1	**226**	55%	1203 (60%)	98	2880

AsymSched

- User-level thread+memory placement manager
- Continuously measures communication
- Decides every second whether threads/memory should be migrated

AsymSched – Measurement

- Reads some hardware counter (data accesses from CPU to node)
- No counter for CPU to CPU available
- Assumes for decision making:
  - Threads on same node share data
  - Between nodes with 'high' communication threads of same application share data.

AsymSched – Decision

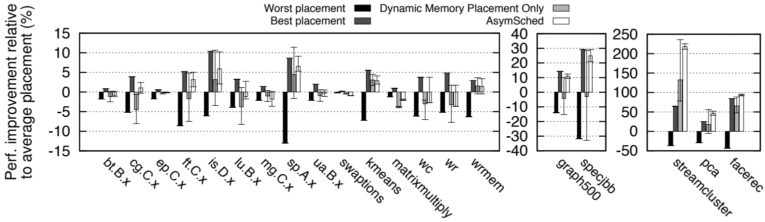
- Puts threads of same application that share data into clusters.
- Each cluster gets weight $C_w = \log(\#\text{remote memory accesses})$.
- For each placement (mapping of clusters to nodes), compute $P_w = \sum_{C \in \text{Clusters}} C_w \cdot (\text{max bandwidth for } C)$.
- Select placements whose $P_w \geq 90\%$ of maximal P_w. Of those choose that with least page migrations.
- If cost for memory migration (assuming 0.3s per GB) is too high, do not apply placement.
- Because of symmetry, not all placements need to be tested. Also "obviously bad" placement are ignored.

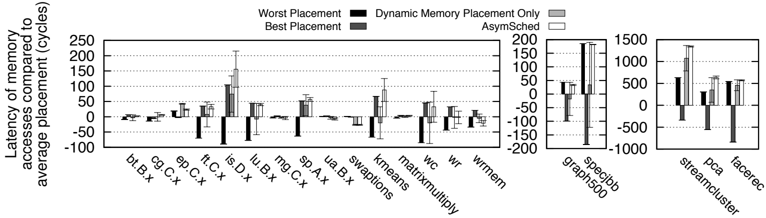
AsymSched – Migration

- Uses *dynamic* (lazy) *migration*.
- If after 2 seconds $>$ 90% of accesses go to old node, do full migration.
- Full migration uses special system call, that is faster than `migrate_pages`, because it stops the application and needs less locks.

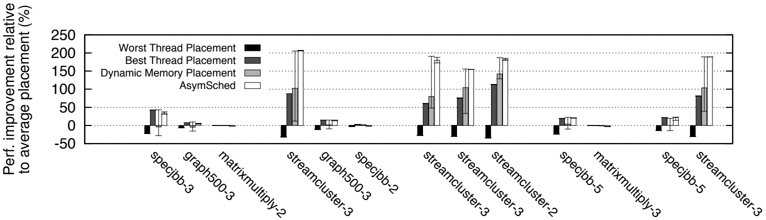
	cg.B	ft.C	is.D	sp.A	streamcluster	graph500	specJBB
Migrated memory (GB)	0.17	2.5	20	0.1	0.15	0.3	10
Average time - Linux syscall (ms)	860	12700	101000	490	750	1500	50500
Average time - fast migration (ms)	51	380	3050	30	45	90	1500

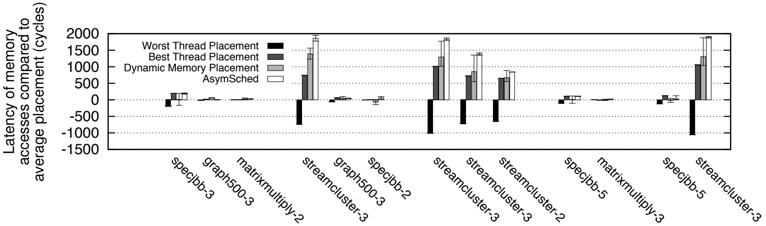
Evaluation – 1 application on 3 nodes





Evaluation – 3 applications





Discussion

- What's the matter with memory migration?
- How well would this work without the magic constants?
- What if #threads is not a multiple of #cores in NUMA-domain?