TABARNAC: Visualizing and Resolving Memory Access Issues on NUMA Architectures

David Beniamine   Matthias Diener   Guillaume Huard
Philippe O. A. Navaux

November 24, 2015

Memory in HPC

- 100's of CPU cycles per memory access.
- Cache hierarchy.
- NUMA machines.
- GPGPU / Xeon Phi ...

Memory in HPC

- 100's of CPU cycles per memory access.
- Cache hierarchy.
- NUMA machines.
- GPGPU / Xeon Phi . . .

Question

How can we analyze and optimize an application's memory usage?

Traditional Analysis tools

- Performance counters [THW10].
- VTune [Rei05].
- HPCTOOLKIT [ABF+10].
- ...

Traditional Analysis tools

- Performance counters [THW10].
- VTune [Rei05].
- HPCTOOLKIT [ABF+10].
- ...

Limits

Do not explain memory issues.

Memory oriented tools

Data and Thread Mapping

- Manual require knowledge.
- Adaptive [DCN13, Lev09]:
  - Collect and use data online.
  - Cannot change memory access pattern.

Memory oriented tools

Data and Thread Mapping

- Manual require knowledge.
- Adaptive [DCN13, Lev09]:
  - Collect and use data online.
  - Cannot change memory access pattern.

Memory profilers [GGR+14, LMC14, LLQ12]

- Based on CPU counters indirect information.
- Sampling (IBS/PEBS etc.) not complete.
- Focus on particular events (remote accesses, cache misses . . .).

Memory oriented tools

Data and Thread Mapping

- Manual require knowledge.
- Adaptive [DCN13, Lev09]:
  - Collect and use data online.
  - Cannot change memory access pattern.

Memory profilers [GGR+14, LMC14, LLQ12]

- Based on CPU counters indirect information.
- Sampling (IBS/PEBS etc.) not complete.
- Focus on particular events (remote accesses, cache misses . . .).

Limits

Not able to give an overview of the memory behavior.

Outline

Outline

Design

Data collection

- Pin (Intel) instrumentation.
- Count number of accesses per page and per threads.
- Retrieve data structures informations (name, size, address).

Design

Data collection

- Pin (Intel) instrumentation.
- Count number of accesses per page and per threads.
- Retrieve data structures informations (name, size, address).

Visualization

- Data oriented visualization.
- Simple yet meaningful plots.
  - Structure size.
  - Number of reads/writes per structure.
  - Accesses distribution by on each structure.
  - First touch distribution by on each structure.
- Explanations about plots' meanings for new users.

Visualizing data structures importance

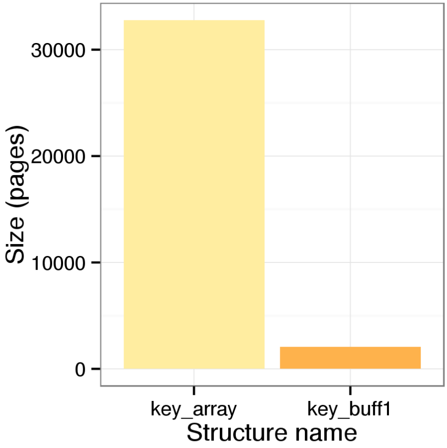

Figure: Comparing structure sizes (number of pages)

Visualizing data structures importance

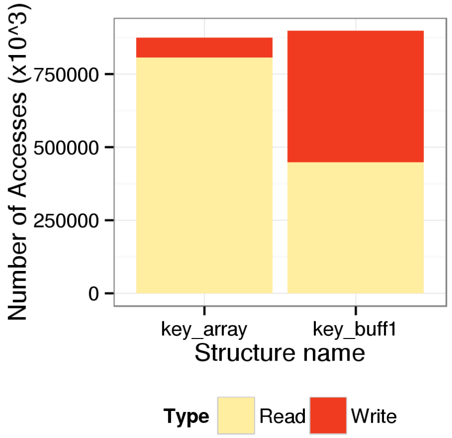

Figure: Number of accesses (reads and writes) by structures

Visualizing memory patterns inside structures

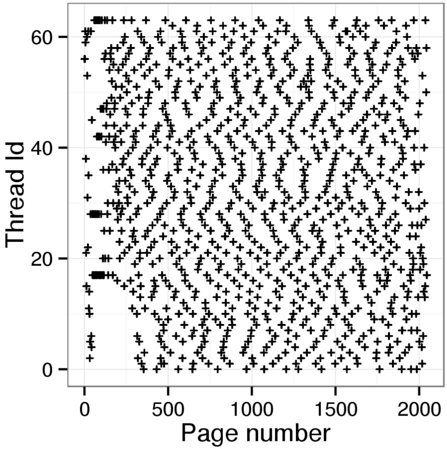

Figure: First touch distribution

Visualizing memory patterns inside structures

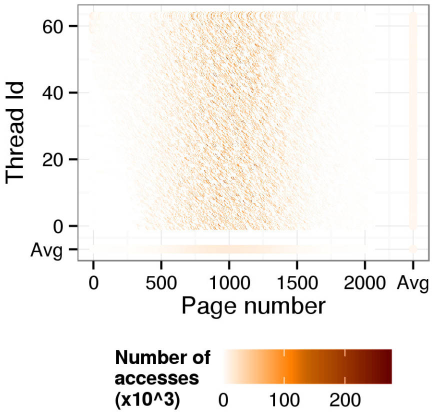

Figure: Access distribution

Outline

IS accesses distribution

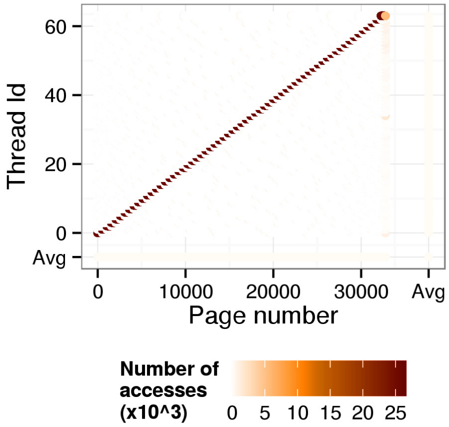

Figure: Access distribution on structure `key_array`

IS accesses distribution

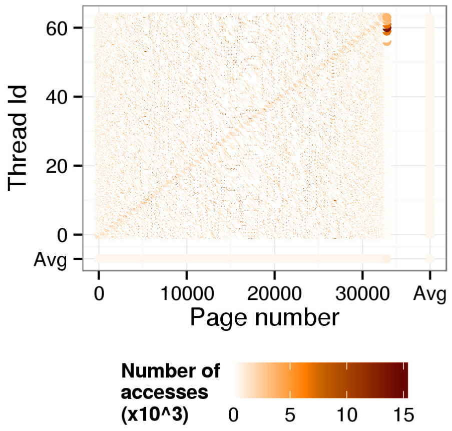

Figure: Access distribution on structure `key_buff2`

IS accesses distribution

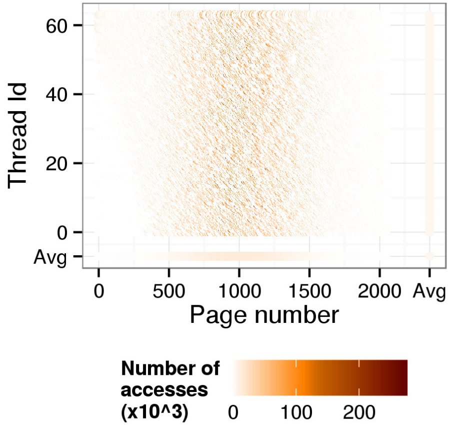

Figure: Access distribution on structure `key_buff1`

What to do now?

What is happening?

- Hot pages intensively accessed by every thread.
- Indirect accesses: `key_buff1[key_buff2[i]]++`.
- Values in `key_buff2` follow a Gaussian distribution.
- Dynamic (OpenMP) thread scheduling (for work balance).

What to do now?

What is happening?

- Hot pages intensively accessed by every thread.
- Indirect accesses: `key_buff1[key_buff2[i]]++`.
- Values in `key_buff2` follow a Gaussian distribution.
- Dynamic (OpenMP) thread scheduling (for work balance).

How to fix this pattern?

- Give each thread hot and cold pages:
  - Split the loop into two parts.
  - Static cyclic distribution of threads.

New accesses distribution

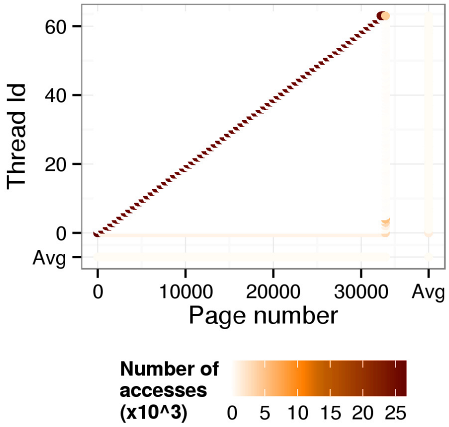

Figure: Modified access distribution on structure `key_array`

New accesses distribution

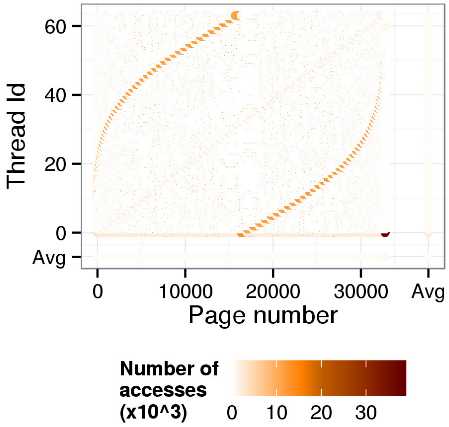

Figure: Modified access distribution on structure `key_buff2`

New accesses distribution

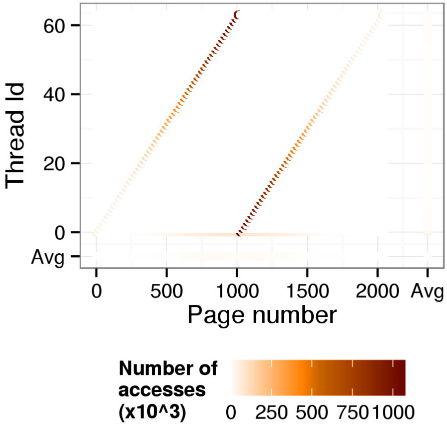

Figure: Modified access distribution on structure `key_buff1`

Methodology

Optimization methods

- Base (OS).
- NUMA balancing (dynamic mapping).
- Interleave (static mapping).

Methodology

Optimization methods

- Base (OS).
- NUMA balancing (dynamic mapping).
- Interleave (static mapping).

Thread distribution

- Dynamic (default).
- Cyclic (don't split the loop, unbalanced accesses).
- Cyclic-split (described earlier + manual mapping).

Results

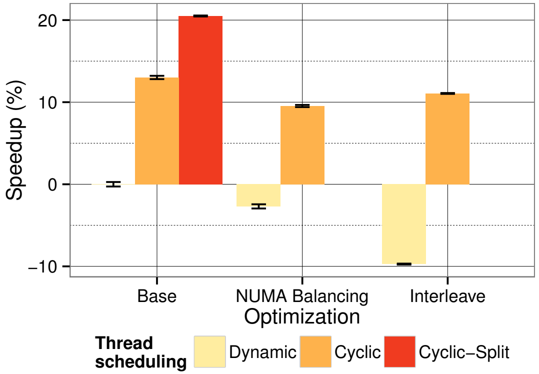

Figure: Evaluation of the modified code

Outline

Contributions

Using efficiently memory is hard

How can we understand an application's memory behavior?

Contributions

Using efficiently memory is hard

How can we understand an application's memory behavior?

TABARNAC (`http://dbeniamine.github.io/Tabarnac/`)

- NUMA oriented.
- Pin-based instrumentation.
- Simple yet meaningful R visualization.
- Optimization hints and explainations for new users.
- Identification of important data structures.
- Sharing patterns inside data structures.

Future work

Study case and optimization

Analyze and study more applications.

Future work

Study case and optimization

Analyze and study more applications.

Visualization and analysis

- Automatic analysis.
- Compute metrics / grade memory patterns.
- Add temporal information.

Questions?

Contact

- `David.Beniamine@Imag.fr`
- `http://moais.imag.fr/membres/david.beniamine/`
- `https://github.com/dbeniamine`

Acknowledgments

Thanks to the CAPES/COFECUB for financing this work.

L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent.
HPCTOOLKIT: tools for performance analysis of optimized parallel programs.
Concurrency and Computation: Practice and Experience, 22(6):685–701, 2010.

M. Diener, E. H. M. Cruz, and P. O. A. Navaux.
Communication-Based Mapping Using Shared Pages.
In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 700–711, May 2013.

Alfredo Giménez, Todd Gamblin, Barry Rountree, Abhinav Bhatele, Ilir Jusufi, Peer-Timo Bremer, and Bernd Hamann.
Dissecting On-Node Memory Access Performance: A Semantic Approach.

In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 166–176. IEEE Press, 2014.

David Levinthal.
Performance Analysis Guide for Intel® Core^{TM} i7 Processor and Intel® Xeon^{TM} 5500 processors.
Technical report, 2009.

Renaud Lachaize, Baptiste Lepers, and Vivien Quema.
MemProf: A Memory Profiler for NUMA Multicore Systems.
In *USENIX 2012 Annual Technical Conference (USENIX ATC 12)*, pages 53–64. USENIX, 2012.

Xu Liu and John Mellor-Crummey.
A Tool to Analyze the Performance of Multithreaded Programs on NUMA Architectures.

In *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '14, pages 259–272. ACM, 2014.

James Reinders.
VTune performance analyzer essentials.
Intel Press, 2005.

J. Treibig, G. Hager, and G. Wellein.
LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments.
In *Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*, 2010.

Machines

CPU		Vendor		Model	
	Turing	Intel		Xeon X7550	
	Idfreeze	AMD		Opteron 6174	
System totals		Nodes	Threads	Freq	Memory
	Turing	4	64	2.00 Ghz	128 Gib
	Idfreeze	8	48	2.20 Ghz	256 Gib
Per node		Cores	Threads	L3 Cache	Memory
	Turing	8	16	18 Mib	32 Gib
	Idfreeze	6	6	12 Mib	32 Gib

Data collection overhead

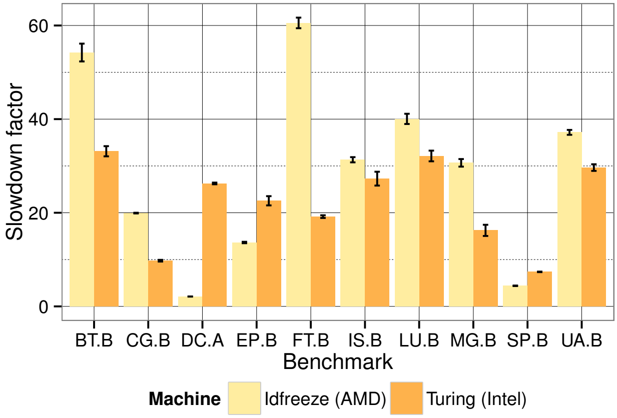

Figure: Overhead of Tabarnac on the NPB on a 64 cores NUMA machine