SnuMAP

An OS-level multi-core applications profiler
https://github.com/SnuMAP/SnuMAP

Computer Systems and Platforms Laboratory
School of Computer Science and Engineering
Seoul National University

Background and Motivation

Multi-core Computing era.

OpenMP®

APACHE hadoop

Intel® Threading Building Blocks

APACHE STORM

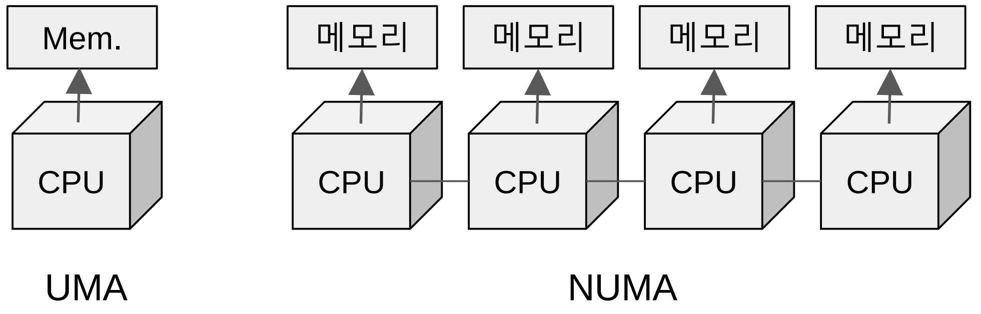
OpenCL

Computing Platforms

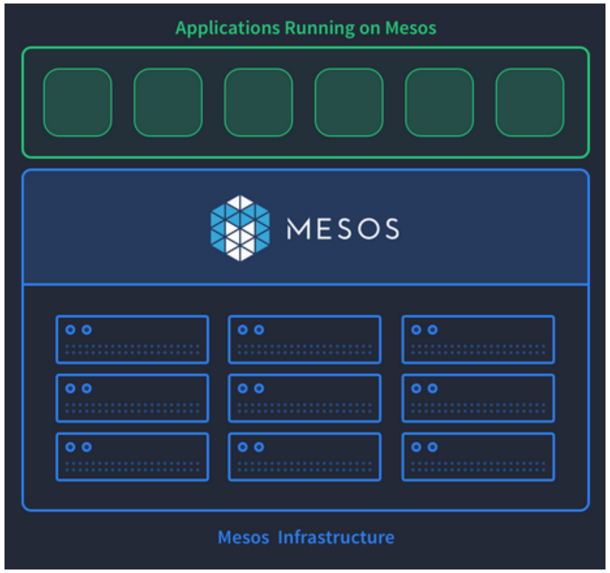
UMA: Uniform Memory Access

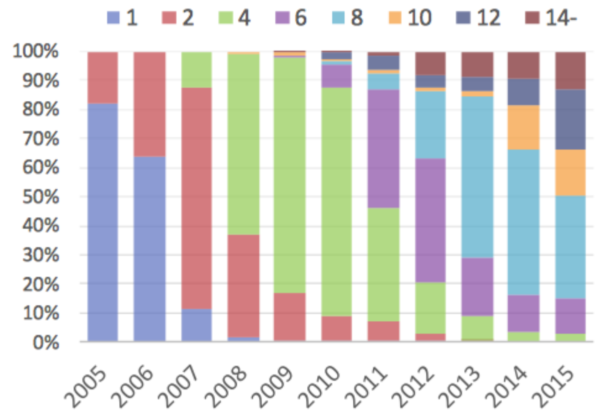
NUMA: Non-Uniform Memory Access

- ❖ Mutli/many-core platforms



- ❖ Data centers and super computing centers





# of cores in a CPU socket in Top 500 supercomputers. (Courtesy of Argonne national lab.)

Are parallel programs efficiently executed in multicores?

❖ **Performance bugs and scheduler bugs**

- Is the program code well-written?
- Are the work-units well-distributed for parallel processing?
- Does the program utilize complex memory hierarchies?
- Does the program run well with the platform’s other workloads?

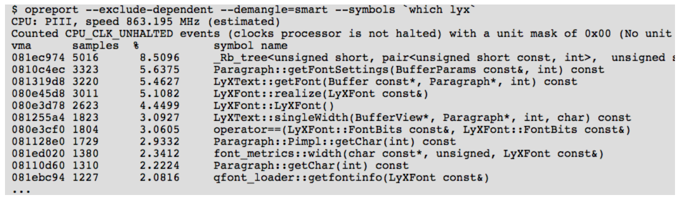
- **Performance bugs and scheduler bugs heavily affect platform’s efficiency**

Tools to find performance bugs

Parallel application profiler from Rice university [BSD3]
Trace profiling and visualization based on a sampling technique to reduce performance degradation.

PIN: application performance analyzer by Intel
provides Instruction-level performance analysis, but there are application performance degradation.

Linux application performance profiler [GPL2]
Performance degradation, and text-based performance information is hard to understand.



```
$ opreport --exclude-dependent --demangle=smart --symbols `which lyx`
CPU: PIII, speed 863.195 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted) with a unit mask of 0x00 (No unit
vma      samples  %           symbol name
081ec974 5016      8.5096     _Rb_tree<unsigned short, pair<unsigned short const, int>,  unsigned s
0810c4ec 3323      5.6375     Paragraph::getFontSettings(BufferParams const&, int) const
081319d8 3220      5.4627     LyXText::getFont(Buffer const*, Paragraph*, int) const
080e45d8 3011      5.1082     LyXFont::realize(LyXFont const&)
080e3d78 2623      4.4499     LyXFont::LyXFont()
081255a4 1823      3.0927     LyXText::singleWidth(BufferView*, Paragraph*, int, char) const
080e3cf0 1804      3.0605     operator==(LyXFont::FontBits const&, LyXFont::FontBits const&)
081128e0 1729      2.9332     Paragraph::Pimpl::getChar(int) const
081ed020 1380      2.3412     font_metrics::width(char const*, unsigned, LyXFont const&)
08110d60 1310      2.2224     Paragraph::getChar(int) const
081ebc94 1227      2.0816     qfont_loader::getfontinfo(LyXFont const&)
...
```

Limitations of existing tools

Application performance degradation; Lack of visualization; Or they assume application’s standalone execution on the platforms

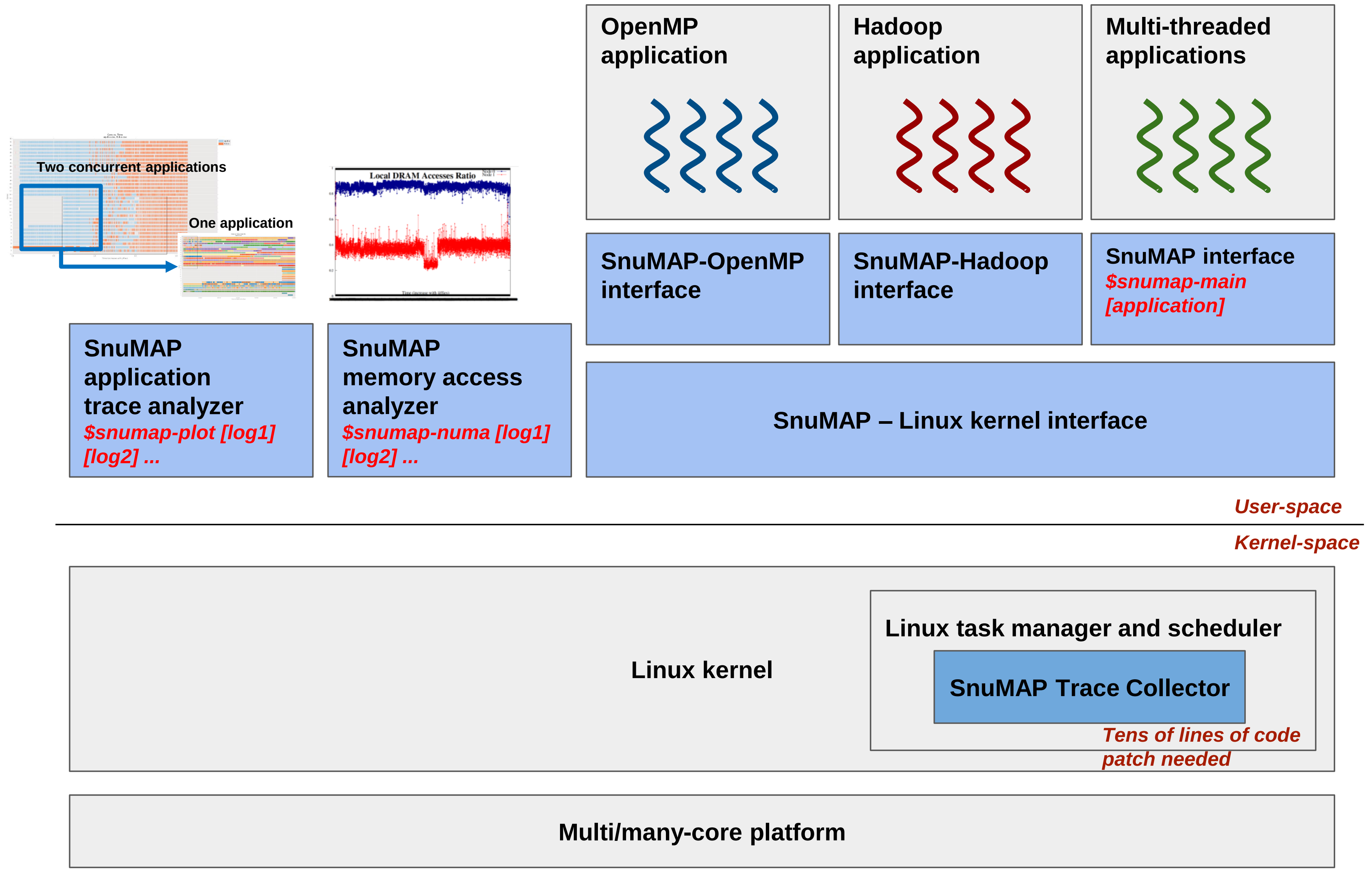
SnuMAP is an OS-level profiler for finding scheduler bugs and performance bugs at the same time, and there is almost no performance degradation.

SnuMAP

SnuMAP

- ❖ **Open-source multi-core application performance profiler**
  - Provide and visualize applications' trace information
  - Provide useful insights for application developers and multi/many-core resource managers
- ❖ **Light weight, no application performance degradation**
  - Analyze trace information in OS-level
  - Need for OS-kernel patch, but independent to the hardware platform
- ❖ **Extensive for parallel application frameworks**
  - Currently, we support Pthread / OpenMP / Hadoop frameworks
  - Need API porting to enable SnuMAP for other parallelization frameworks

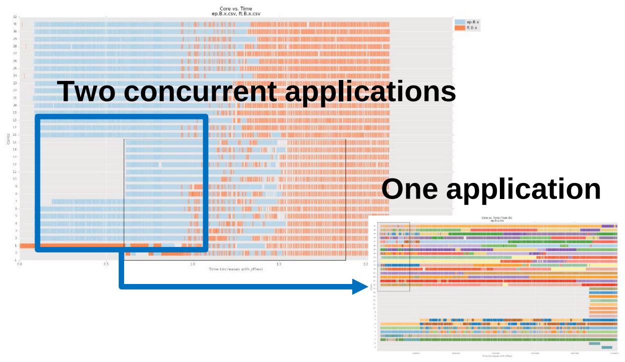
SnuMAP Framework


OpenMP application
Hadoop application
Multi-threaded applications
SnuMAP-OpenMP interface
SnuMAP-Hadoop interface
SnuMAP interface
Two concurrent applications
One application
Local DRAM Accesses Ratio
SnuMAP application trace analyzer
SnuMAP memory access analyzer
SnuMAP – Linux kernel interface
User-space
Kernel-space
Linux kernel
Linux task manager and scheduler
SnuMAP Trace Collector
Tens of lines of code patch needed
Multi/many-core platform

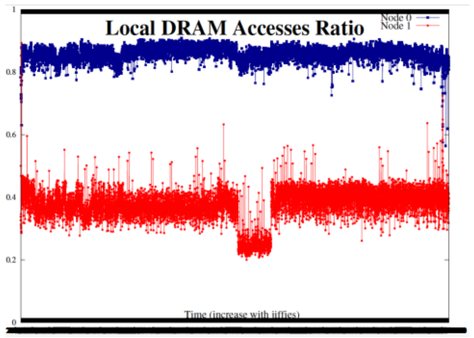
```
$snumap-main [application]
$snumap-plot [log1] [log2] ...
$snumap-numa [log1] [log2] ...
```

Performance Information from SnuMAP

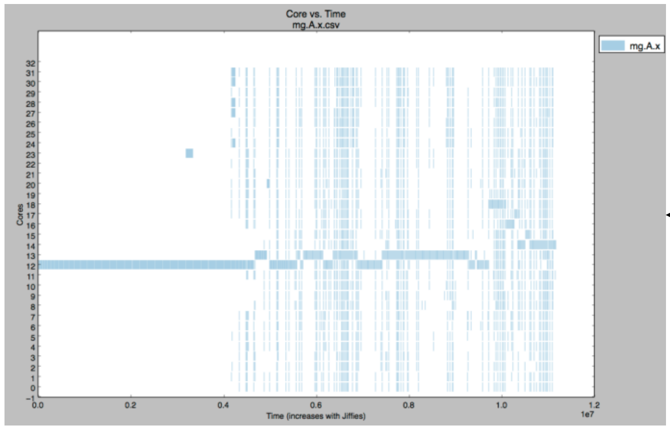
- ❖ **Application trace analysis and visualization on multi-core platforms**



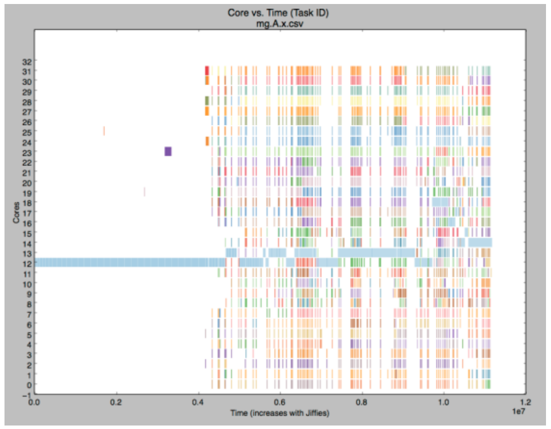
- ❖ **Application memory access pattern in NUMA**



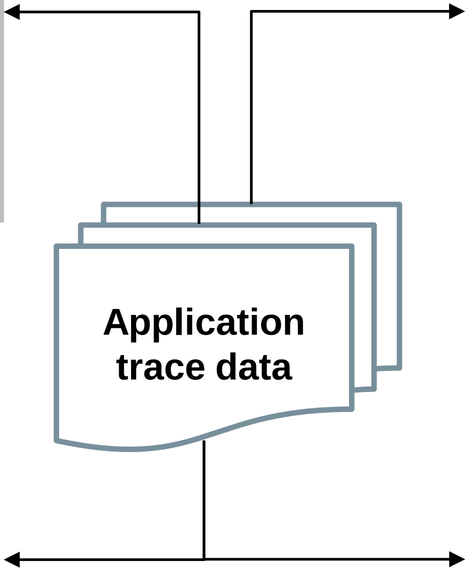
Application Trace Visualization

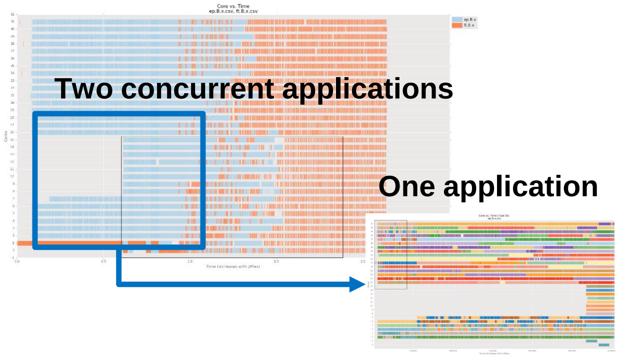


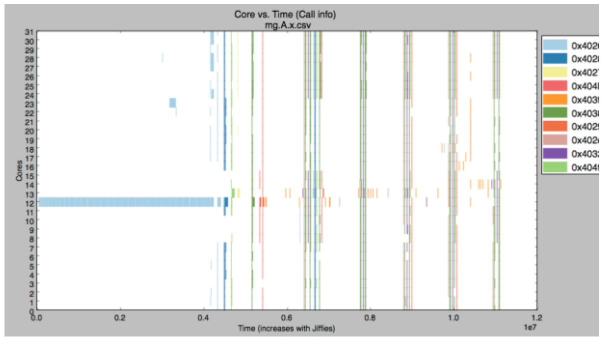
Per-application trace visualization



Per-thread trace visualization





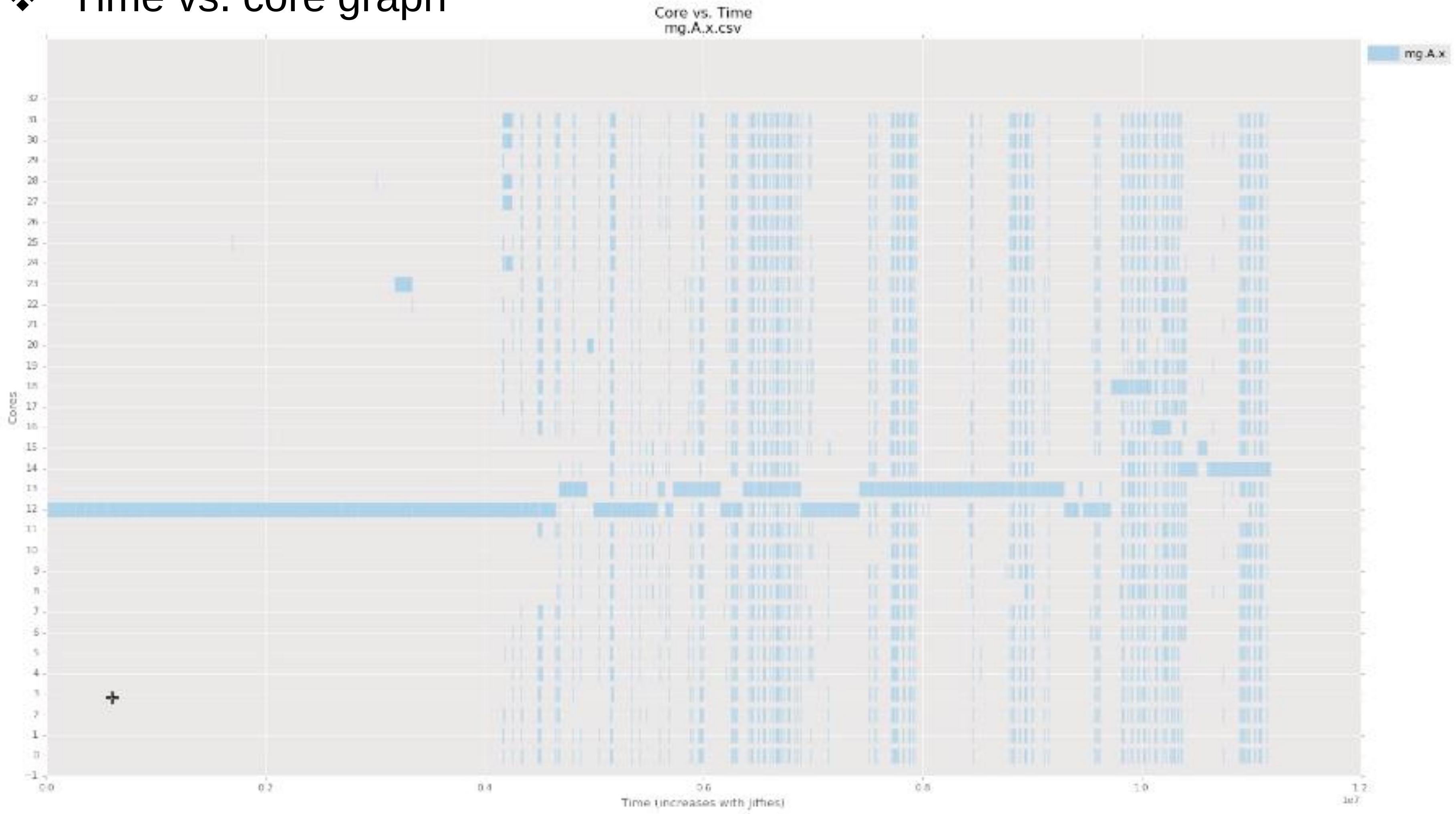


Code section based trace visualization

Multiple concurrent applications trace visualization

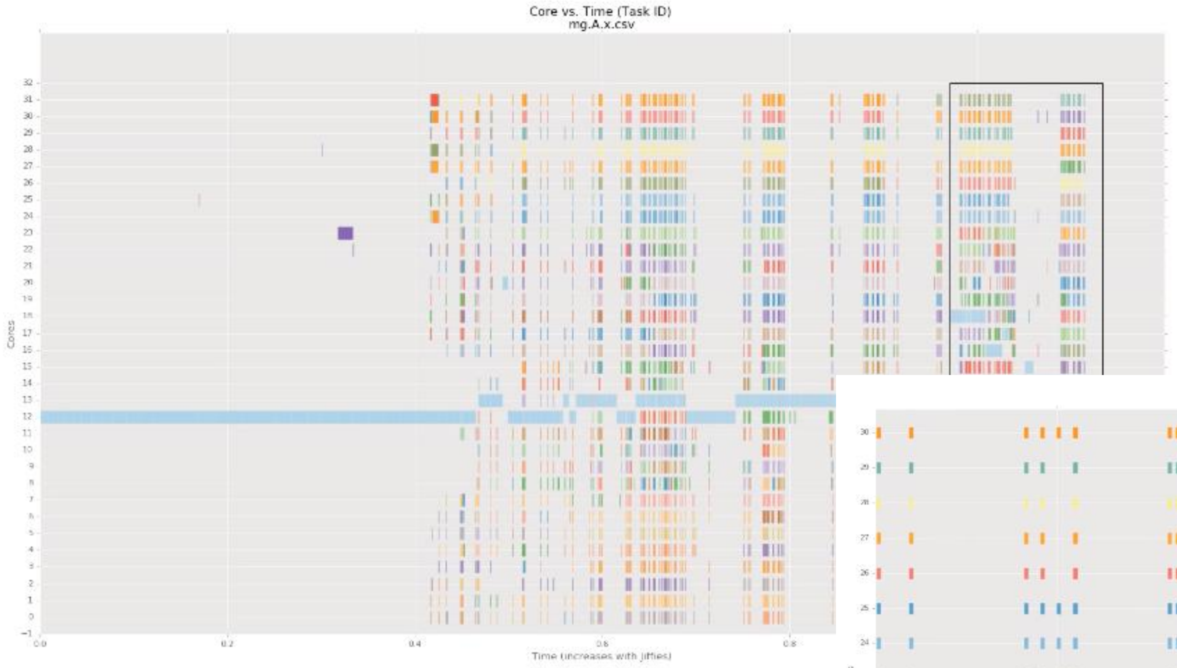
Per-application Trace Info.

- Time vs. core graph

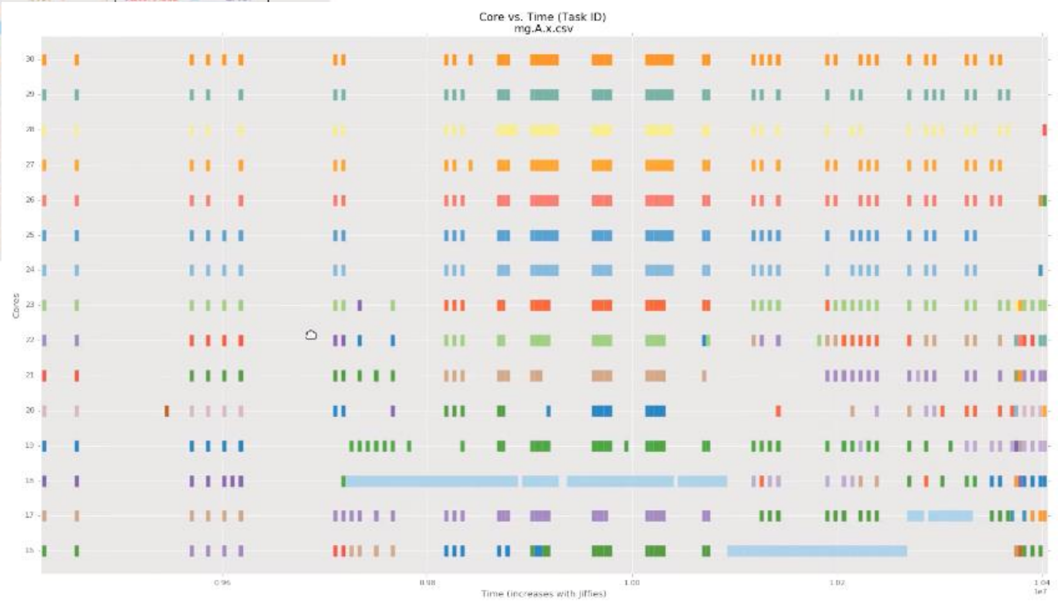


Per-thread Trace Info.

- Threads in an application

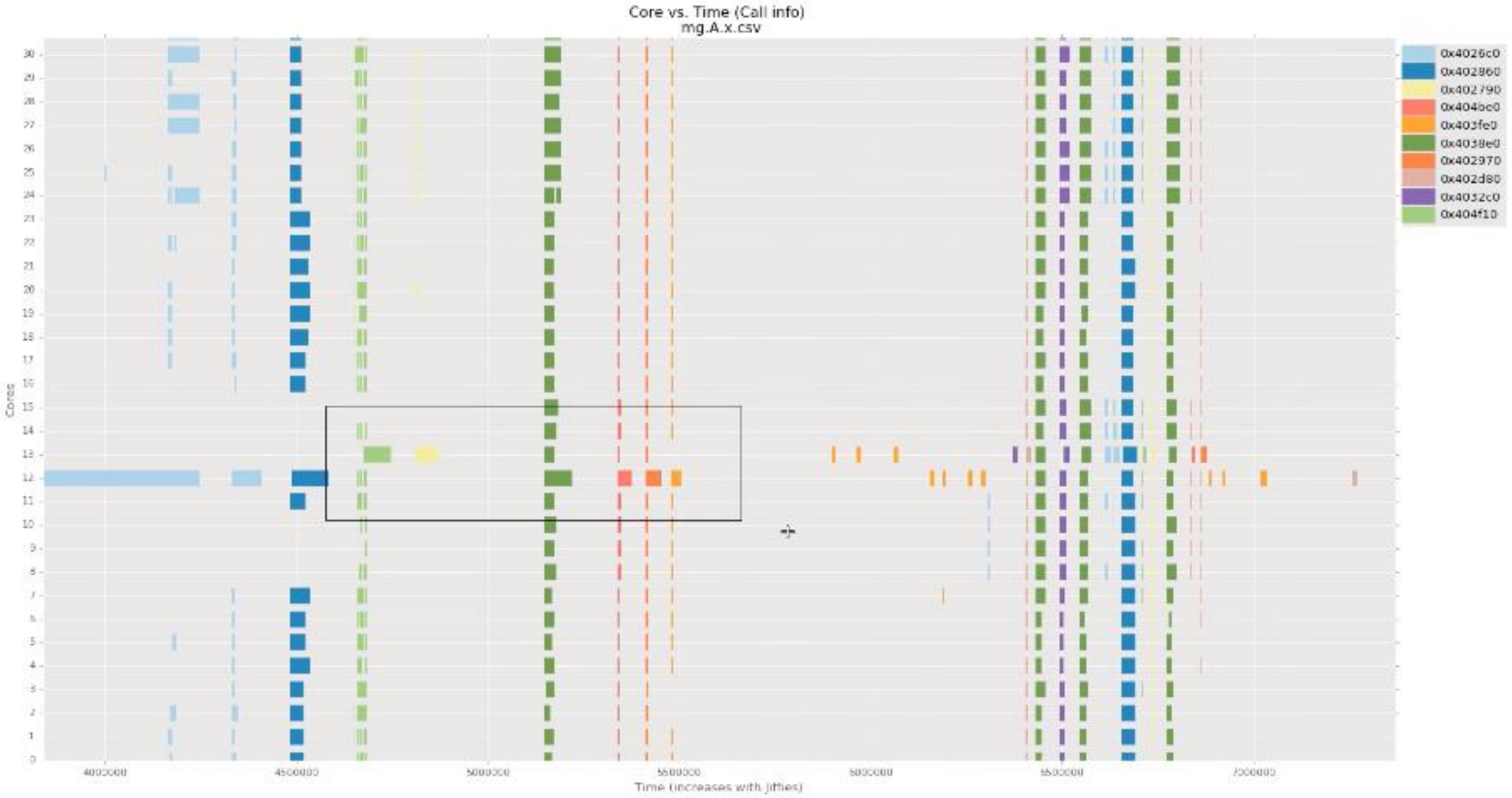


Zoom in



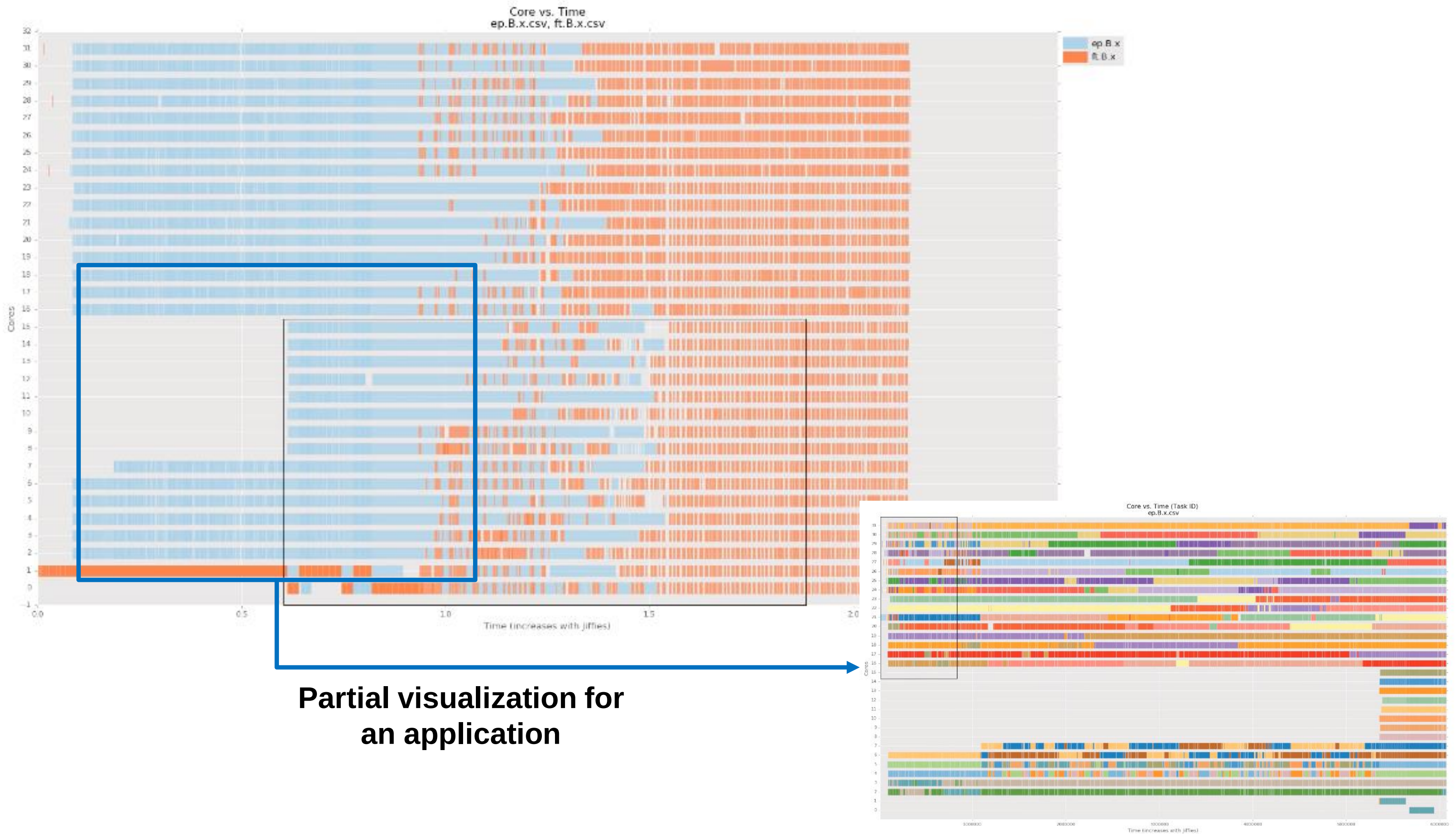
Code Section based Analysis

- Code sections in an application



Concurrent Applications

- ❖ Visualize trace interference between multiple concurrent applications



Partial visualization for an application

Effectiveness of SnuMAP

Through SnuMAP

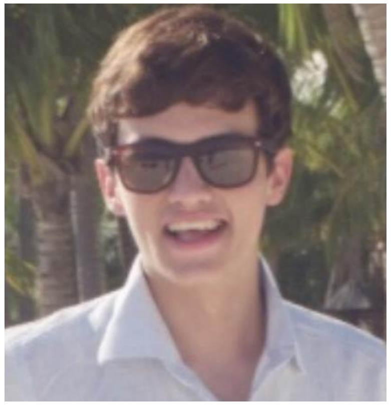
1. **We can find application performance bugs and scheduling bugs that enables application performance tuning or improving resource management scheme.**

2. **We can provide efficient co-scheduling of multiple concurrent applications that improves computing efficiency of multi-core platforms.**

SnuMAP Testbed

- ❖ Multi-core platforms
  - Tested on 64-core and 32-core AMD Opteron serves, and 36-core Tile-Gx36 processor
- ❖ Multi-threaded applications
  - Tested with Pthread applications, OpenMP applications and Hadoop JAVA applications
- ❖ Linux kernel patch
  - Tested on Linux kernel 2.x and 3.x

Contributors


Camilo A. Celis Guzman
(SNU)

Heesik Shin
(SAP Korea)

Younghyun Cho
(SNU)

Surim Oh
(SNU)

Bernhard Egger
(SNU)

Computer Systems and Platforms Laboratory
School of Computer Science and Engineering
Seoul National University