# OmpSs Tutorial: AGENDA

| TIME | TOPIC |
|---|---|
| 13:00 – 14:00 | OmpSs Quick Overview |
| 14:00 – 14:30 | Hands on Exercises |
| 14:30 – 15:00 | *-- Coffee Break --* |
| 15:00 – 16:00 | Fundamentals of OmpSs |
| 16:00 – 17:00 | Hands on Exercises |

www.bsc.es


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# OmpSs Quick Overview

## A practical approach

Xavier Teruel

New York, June 2013

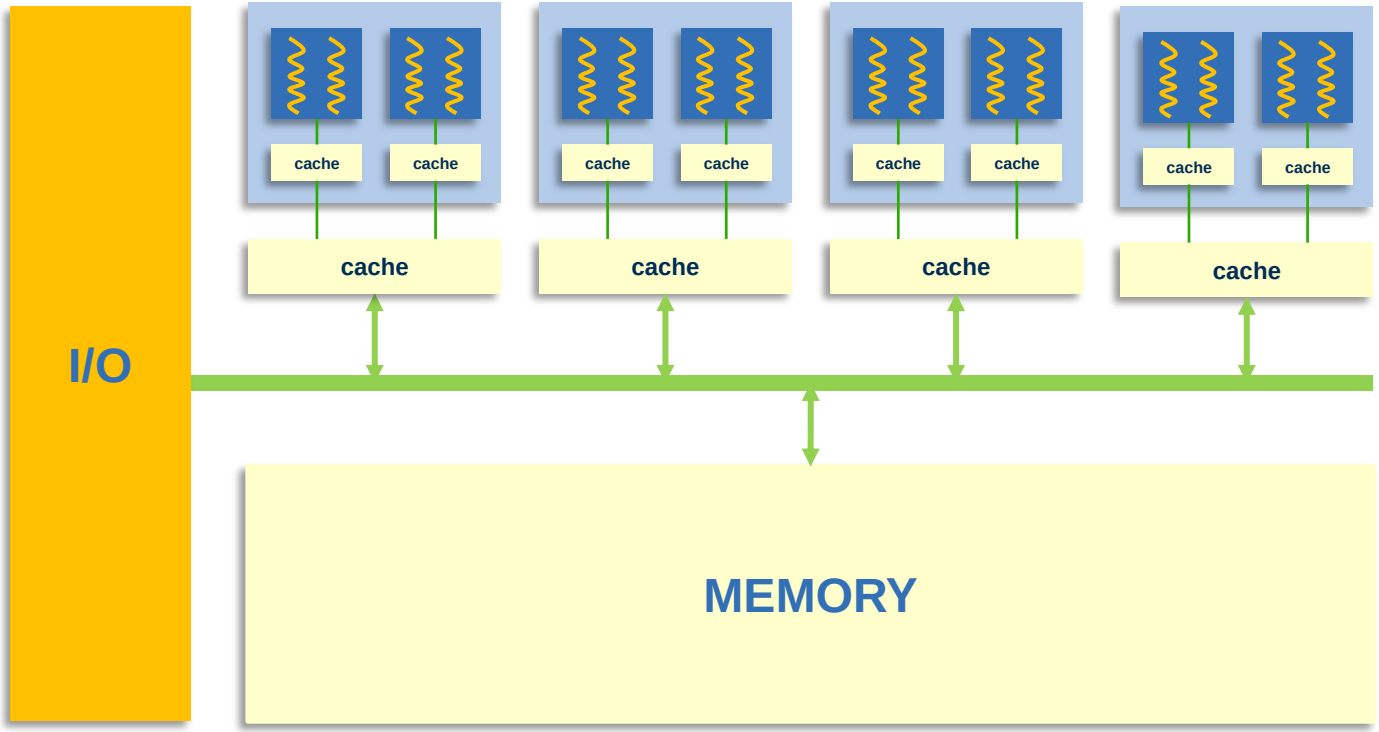
# AGENDA: OmpSs Quick Overview

- High Performance Computing
  - Supercomputers
  - Parallel programming models
- OmpSs Introduction
  - Programming model main features
  - A practical example: Cholesky factorization
- BSC's Implementation
  - Mercurium compiler
  - Nanos++ runtime library
  - Visualization tools: Paraver and Graph
- Hands-on Exercises


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
HIGH PERFORMANCE
COMPUTING

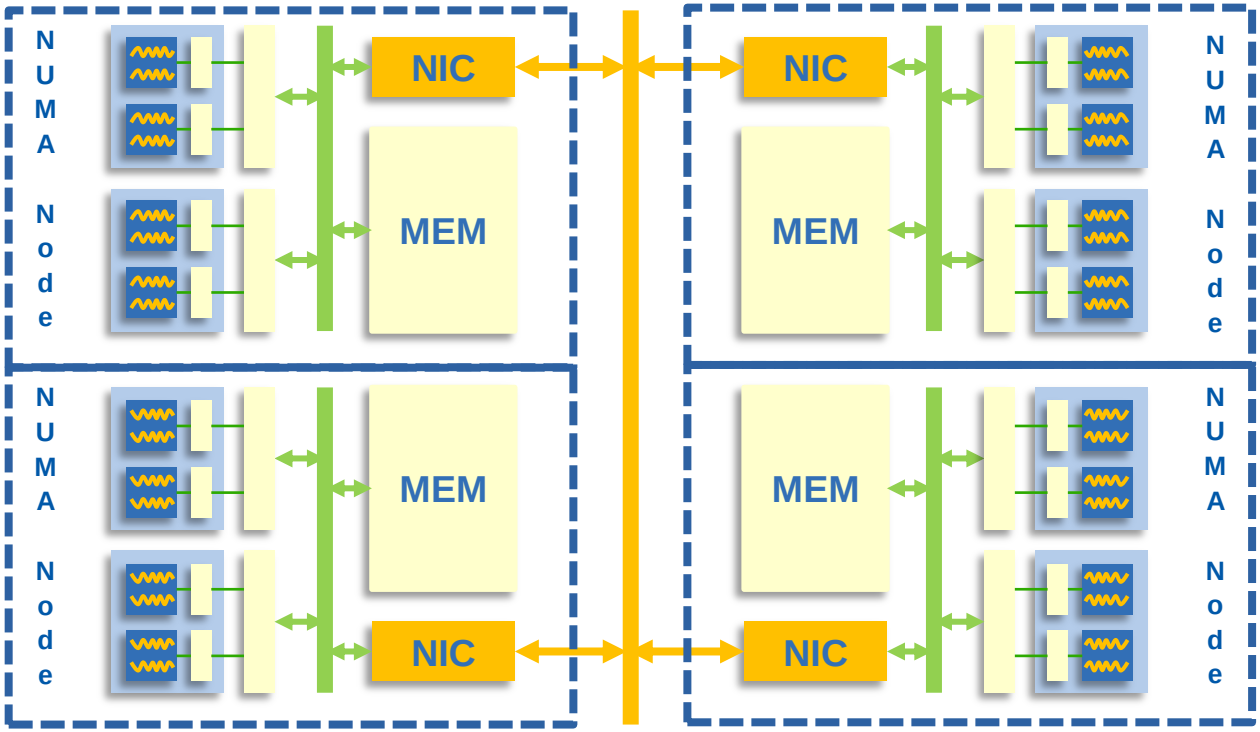
# Supercomputers: Shared Memory Machines

- SMP: Simetric MultiProcessors
  - Shared Memory → Single Address Space
  - Memory hierarchy must keep coherence/consistency access



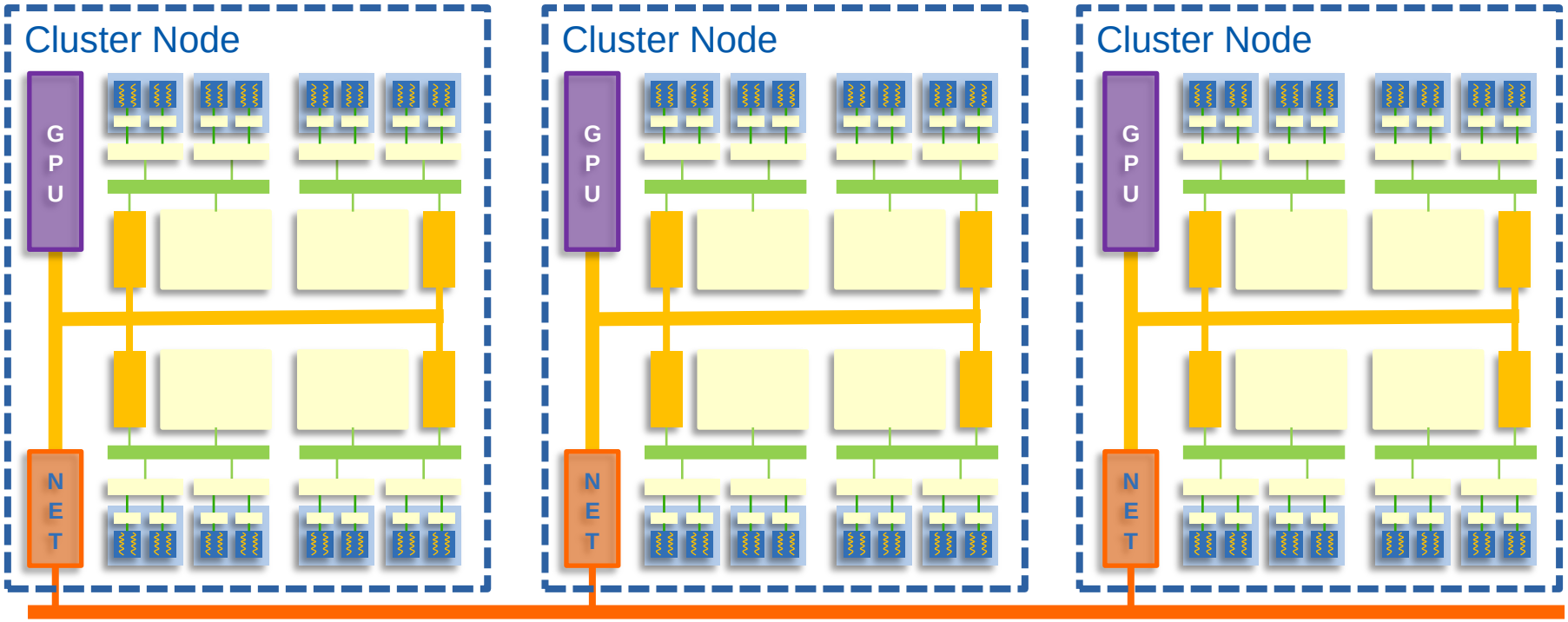
# Supercomputers: Distributed Shared Memory Machines

- NUMA: Non Uniform Memory Access
  - Distributed Shared Memory → Single Address Space
  - Local/Remote memory access → Data locality aware programming



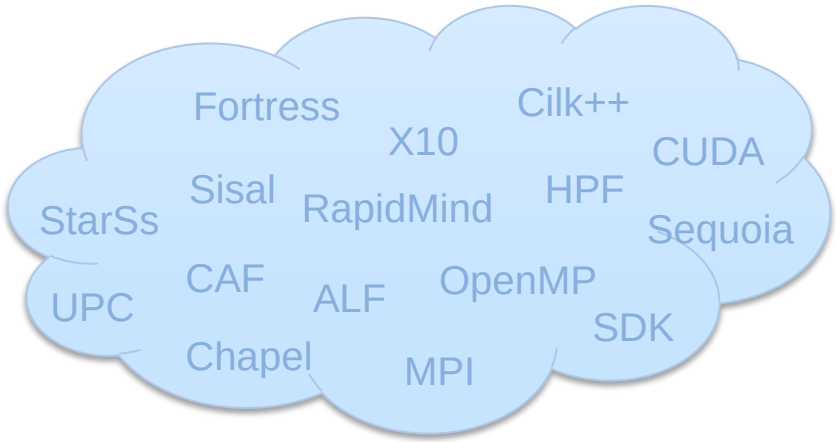
# Supercomputers: Cluster Machines

- SM or DSM machines interconnected
  - Distributed Memory → Multiple Address Spaces
  - Communication through interconnection network
- Usually allows multiple levels of parallelism



# Parallel Programming Models

- Traditional programming models
  - Message passing (MPI)
  - OpenMP
  - Hybrid MPI/OpenMP
- Heterogeneity
  - CUDA
  - OpenCL
  - ALF
  - RapidMind




- New approaches
  - Partitioned Global Address Space (PGAS) programming models
    - UPC, X10, Chapel
- ...

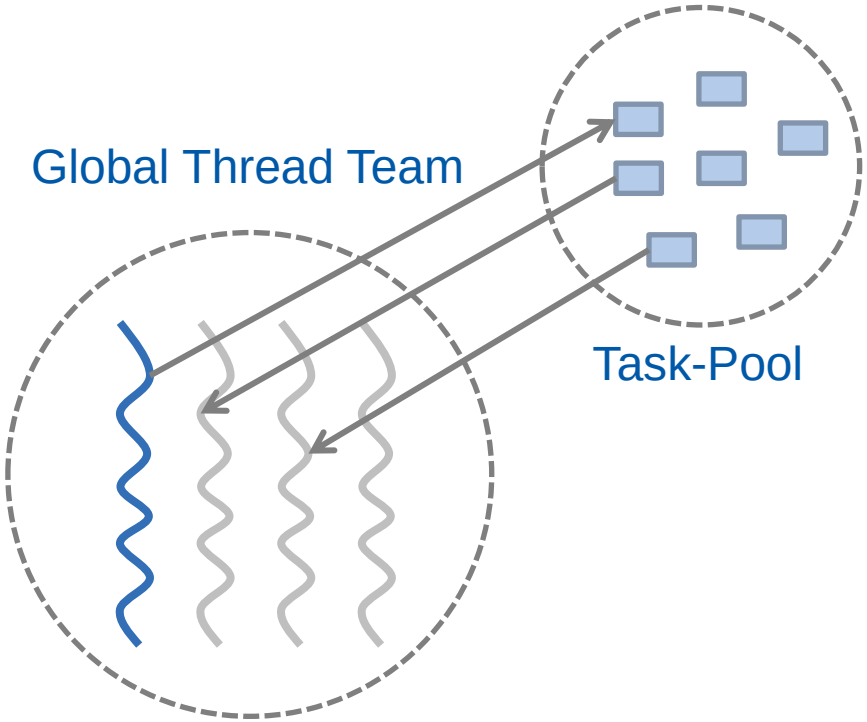

# OMPSS INTRODUCTION

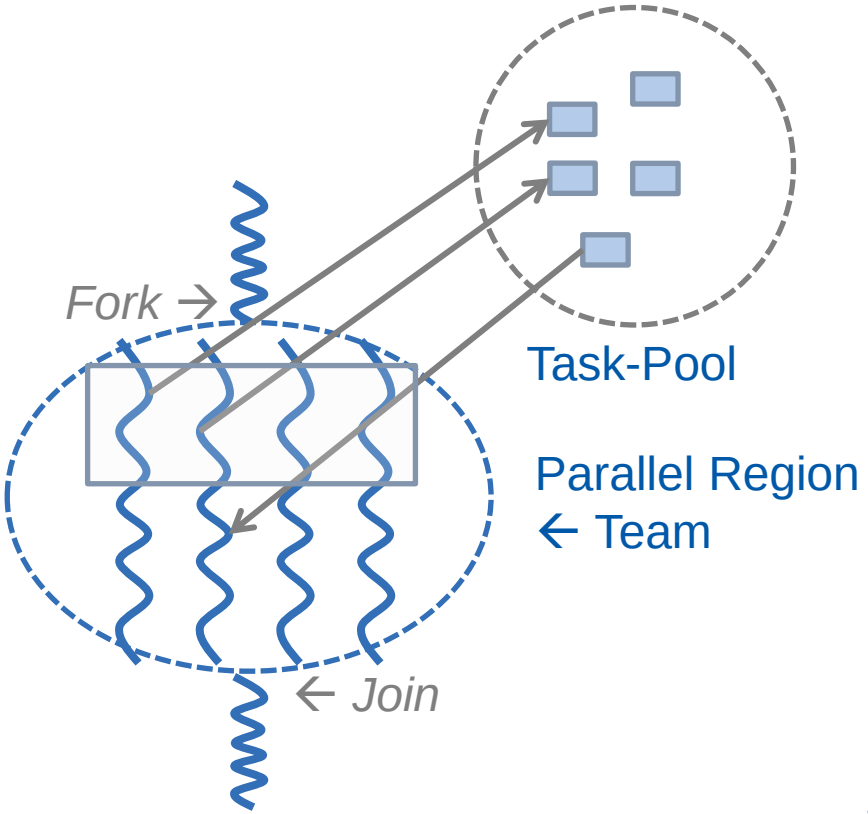
# OmpSs Introduction ~ 1

- Parallel Programming Model
  - Build on existing standard: OpenMP
  - Directive based to keep a serial version
  - Targeting: SMP, clusters and accelerator devices
  - Developed in Barcelona Supercomputing Center (BSC)
    - Mercurium source-to-source compiler
    - Nanos++ runtime system
- Where it comes from (a bit of history)
  - BSC had two working lines for several years
    - OpenMP Extensions: Dynamic Sections, OpenMP Tasking prototype
    - StarSs: Asynchronous Task Parallelism Ideas
  - OmpSs is our effort to fold them together

# OmpSs: Execution Model

- Persistent global thread team (whole execution)
  - All threads are part of the global thread team
  - Only one executes main program / all cooperates in task execution
- OmpSs execution model vs. OpenMP execution model

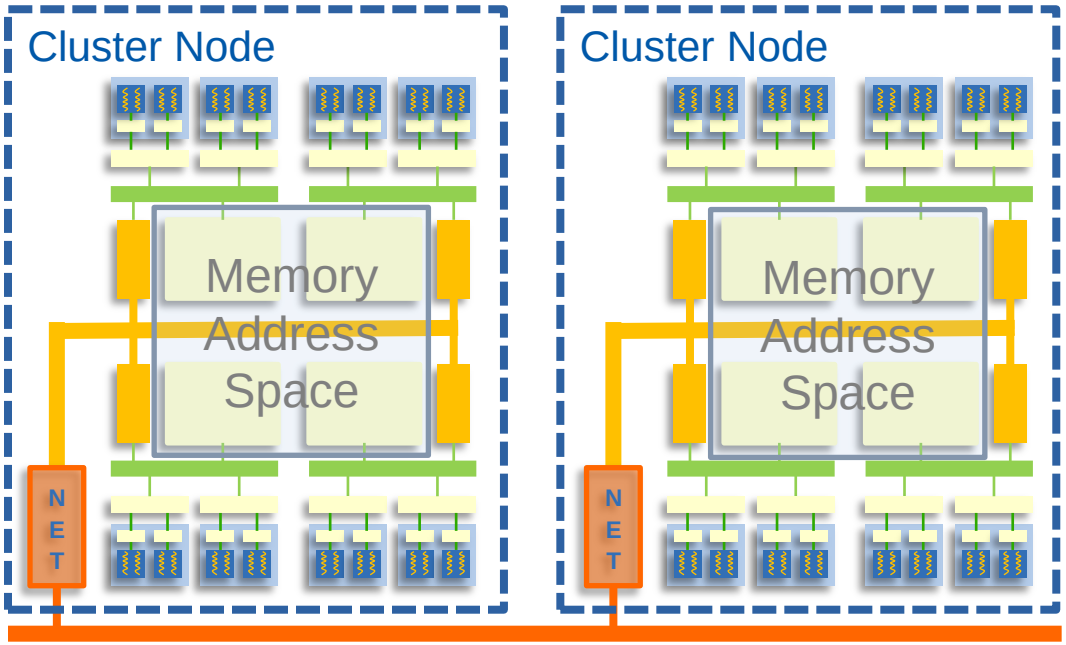


# OmpsSs: Memory Model

- Copy clauses allow to work with Multiple Address Spaces
  - In between cluster nodes and from/to host to/from device
  - Programmer has the perception of having a single address space
  - Runtime library will keep consistency in between memories

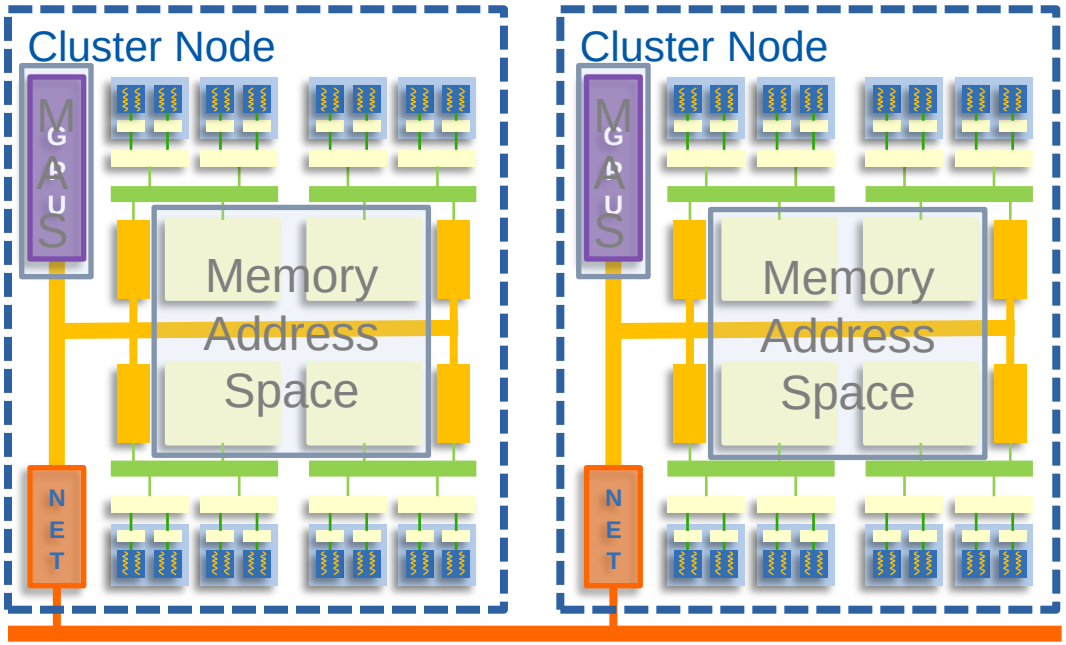
```
int A[SIZE];

#pragma omp target device (smp) \
  copy_out([SIZE] A)
#pragma omp task
  matrix_initialization(A);

#pragma omp taskwait

#pragma omp target device (smp) \
  copy_inout([SIZE]A)
#pragma omp task
  matrix_increment(A);
```



# OmpSs: Heterogeneity Support

- Compiler tool-chain allow to work with heterogeneous code
  - Working with multiple devices architectures
  - Also with multiples implementation of the same function

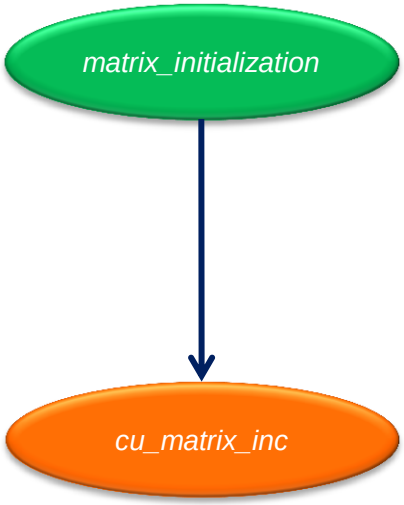
```
int A[SIZE];

#pragma omp target device (smp) \
  copy_out([SIZE] A)
#pragma omp task
  matrix_initialization(A);

#pragma omp taskwait

#pragma omp target device (cuda) \
  copy_inout([SIZE]A)
#pragma omp task
{
  cu_matrix_inc<<<Size,1>>>(A);
}
```



# OmpSs: Asynchronous Data-Flow Execution

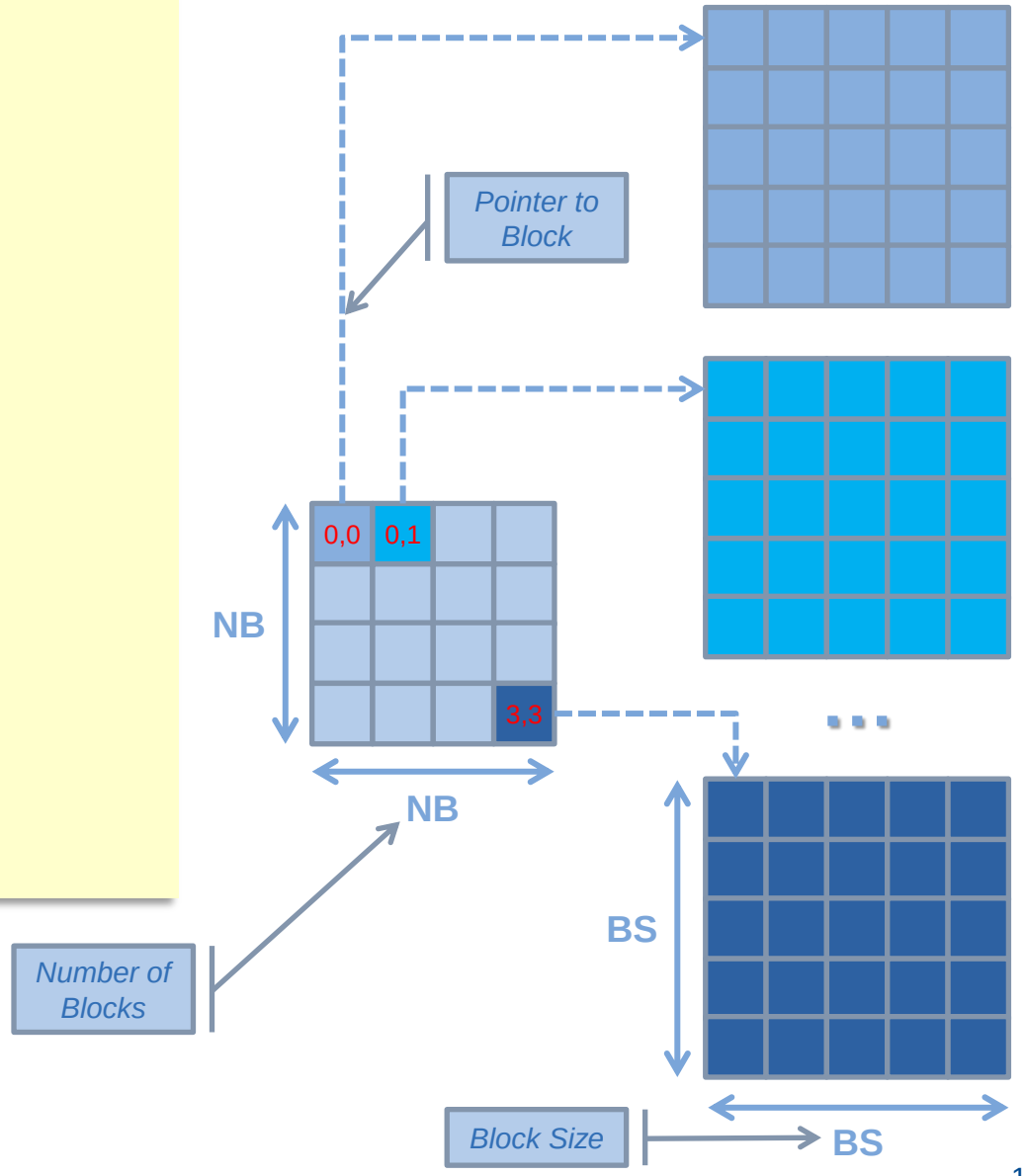
- Dependence clauses allow to remove synch directives
  - Runtime library will compute dependences for us

```
int A[SIZE];

#pragma omp target device (smp) \
  copy_out([SIZE] A)
#pragma omp task out(A)
  matrix_initialization(A);

#pragma omp taskwait

#pragma omp target device (cuda) \
  copy_inout([SIZE]A)
#pragma omp task inout(A)
{
  cu_matrix_inc<<<Size,1>>>(A);
}
```



# Cholesky Factorization (introduction)

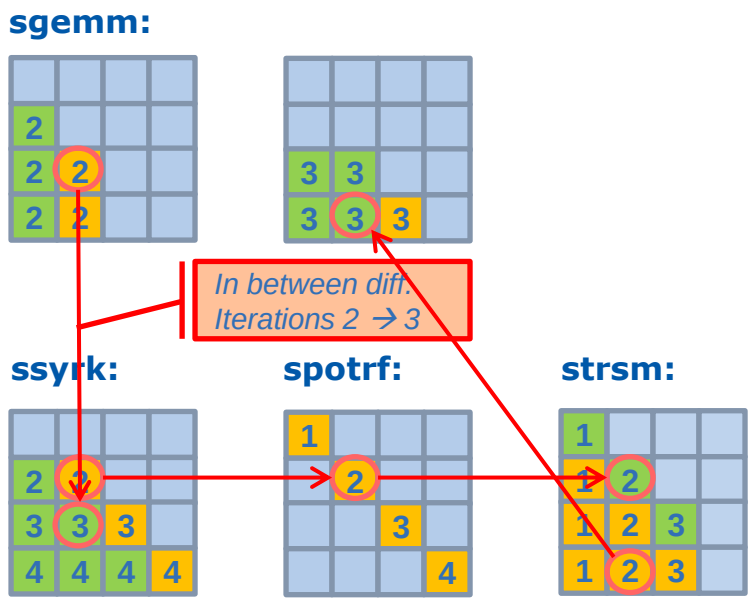
```
for (j = 0; j < NB; j++) {
  for (k = 0; k < j; k++)
    for (i = j+1; i < NB; i++) {

      sgemm(A[i][k], A[j][k], A[i][j]);
  }
  for (i = 0; i < j; i++) {

    ssyrk( A[j][i], A[j][j]);
  }


  spotrf( A[j][j]);

  for (i = j+1; i < NB; i++) {

    strsm( A[j][j], A[i][j]);
  }
}
```



# Cholesky Factorization (explicit synchronization)

```
for (j = 0; j < NB; j++) {
  for (k = 0; k < j; k++)
    for (i = j+1; i < NB; i++) {
#pragma omp task
      sgemm (A[i][k], A[j][k], A[i][j]);
  }
  for (i = 0; i < j; i++) {
#pragma omp task
    ssyrk( A[j][i], A[j][j]);
  }
#pragma omp taskwait
#pragma omp task
  spotrf( A[j][j]);
#pragma omp taskwait
  for (i = j+1; i < NB; i++) {
#pragma omp task
    strsm( A[j][j], A[i][j]);
  }
#pragma omp taskwait
}
```

- In parallel/single construct
- Kernel op's patterns




- Other approaches
  - Using parallel/worksharings
  - Barriers still are needed

# Cholesky Factorization (data-flow synchronization)

```
for (j = 0; j < NB; j++) {
  for (k = 0; k < j; k++)
    for (i = j+1; i < NB; i++) {
#pragma omp task in(A[i][k], A[j][k]) inout(A[i][j]
      sgemm(A[i][k], A[j][k], A[i][j]);
  }
  for (i = 0; i < j; i++) {
#pragma omp task in(A[j][i]) inout(A[j][j])
    ssyrk( A[j][i], A[j][j]);
  }

#pragma omp task inout(A[j][j])
  spotrf( A[j][j]);

  for (i = j+1; i < NB; i++) {
#pragma omp task in(A[j][j]) inout(A[i][j]
    strsm( A[j][j], A[i][j]);
  }
}
```

- In parallel/single construct
- No parallel discussion
  - Just specify memory usage
  - Runtime will compute dependences
- No need of taskwait
  - Dependences P2P

# Cholesky Factorization (task creation, with NB=4)

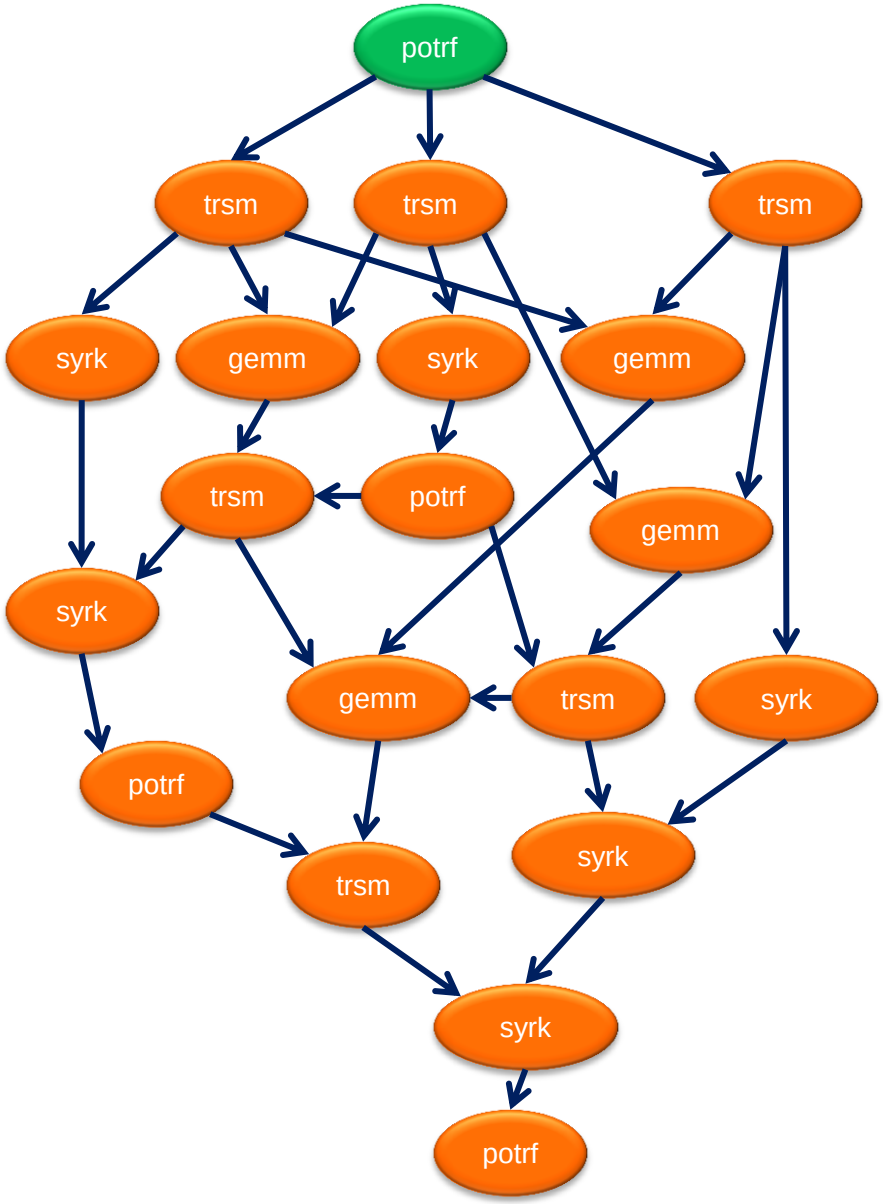
```
for (j = 0; j < NB; j++) {
  for (k = 0; k < j; k++)
    for (i = j+1; i < NB; i++) {
#pragma omp task in(A[i][k], A[j][k]) inout(A[i][j])
      sgemm(A[i][k], A[j][k], A[i][j]);
  }
  for (i = 0; i < j; i++) {
#pragma omp task in(A[j][i]) inout(A[j][j])
    ssyrk( A[j][i], A[j][j]);
  }

#pragma omp task inout(A[j][j])
  spotrf( A[j][j]);

  for (i = j+1; i < NB; i++) {
#pragma omp task in(A[j][j]) inout(A[i][j])
    strsm( A[j][j], A[i][j]);
  }
}
```

j == 4

A[2][0] A[1][0]
A[3][0] A[1][0]

A[0][0]

A[1][0]
A[2][0]
A[3][0]

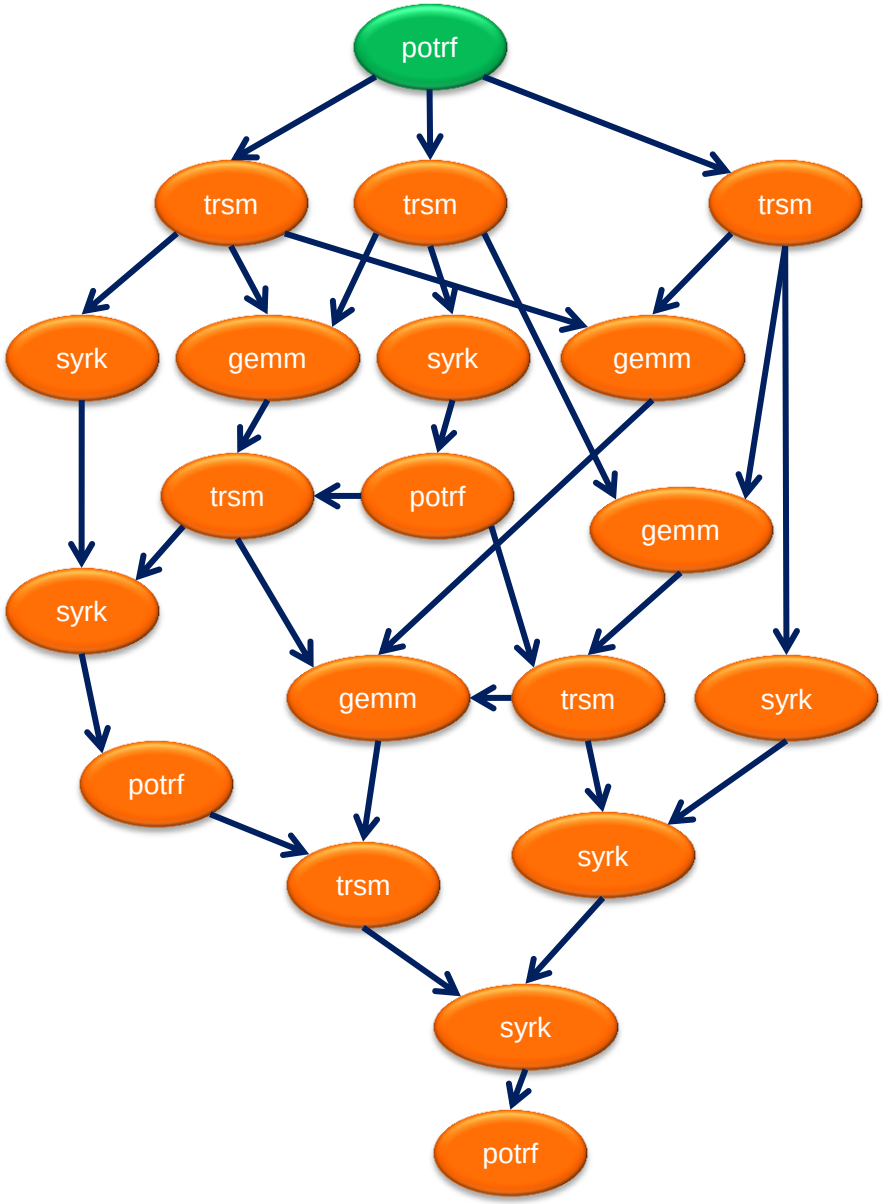



ready queue → potrf

in execution → main

# Cholesky Factorization (task execution, with NB=4)

```
for (j = 0; j < NB; j++) {
  for (k = 0; k < j; k++)
    for (i = j+1; i < NB; i++) {
#pragma omp task in(A[i][k], A[j][k]) inout(A[i][j])
      sgemm(A[i][k], A[j][k], A[i][j]);
  }
  for (i = 0; i < j; i++) {
#pragma omp task in(A[j][i]) inout(A[j][j])
    ssyrk( A[j][i], A[j][j]);
  }

#pragma omp task inout(A[j][j])
  spotrf( A[j][j]);

  for (i = j+1; i < NB; i++) {
#pragma omp task in(A[j][j]) inout(A[i][j])
    strsm( A[j][j], A[i][j]);
  }
}
```



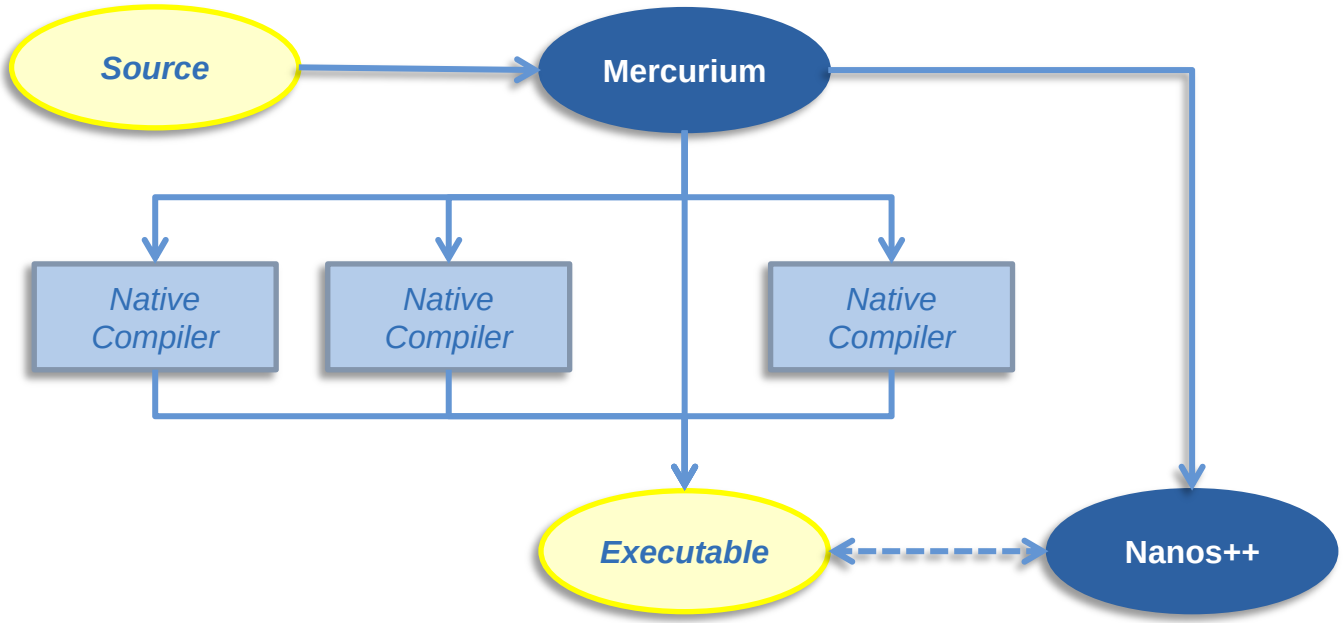

ready queue → potrf trsm trsm

in execution →


# OMPSS IMPLEMENTATION

# OmpSs Implementation

- Mercurium Compiler
  - Source to source compiler: transform OmpSs directives to runtime calls
- Nanos++ RTL
  - Implement runtime services: create/execute tasks, synchronization, comp. dependences, mem. consistency,…



# Compiling OmpSs Applications (Mercurium Options) ~ 1

## Multiple front-ends

| Mercurium | Language |
|---|---|
| mcc | C |
| mcxx | C++ |
| mnvcc | CUDA & C |
| mnvcxx | CUDA & C++ |
| mfc | Fortran |

## Common Flags

| Flag | Description |
|---|---|
| `-c` | Compile only |
| `-o` | Output file name |
| `-g` | Add debug information |

## Common Flags (cont)

| Flag | Description |
|---|---|
| `-IPATH` | Search includes in… |
| `-LPATH` | Search libraries in… |
| `-lname` | Link library name |
| `-DSYM` | Define symbol SYM |
| `--version` | Compiler version |
| `--verbose` | Verbose mode |
| `--help` | Help information |

# Compiling OmpSs Applications (Mercurium Options) ~ 2

## Nanos++ Flags

| Flag | Description |
|---|---|
| `--instrument` | Instrumentation version |
| `--debug` | Debug version |

## Native P/C/L

| Flag | Description |
|---|---|
| `-Wp,flags` | Preprocessor flags |
| `-Wn,flags` | Compiler flags |
| `-Wl,flags` | Linker Flags |

## Intermediate Code

| Flag | Description |
|---|---|
| `-k` | Keep them |
| `--pass-through` | Use It |

## Programming Model

| Flag | Description |
|---|---|
| `--ompss` | OmpSs |
| `--nanox` | OpenMP |

## Compilation using ompss and instrumented version:

```
$ mcc --ompss --instrument -o cholesky cholesky.c
```

# Executing OmpSs Applications (Nanos++ Options) ~ 1

- Nanos++ options → **NX_ARGS="options"**

| Option | Description |
|---|---|
| --threads=n | Defines the number of threads used by the process |
| --disable-yield | Disables thread yielding to OS |
| --spins=n | Number of spins before sleep when idle |
| --sleeps=n | Number of sleeps before yielding |
| --sleep-time=nsecs | Sleep time before spinning again |
| --disable-binding | Do not bind threads to CPUs |
| --binding-start | Initial CPU where start binding |
| --binding-stride | Stride between bound CPUs |

- Execution using *4* threads and *disable-yield* option:

```
$ NX_ARGS="--threads=4 --disable-yield"  .\cholesky  4096 256
```

# Executing OmpSs Applications (Nanos++ Options) ~ 2

- Nanos++ plugins → **NX_ARGS="options"**

| Option | Description |
|---|---|
| --schedule=name | Defines the scheduler policy:<br>- bf → breadht first scheduler<br>- wf → work first scheduler […] |
| --throttle=name | Defines throttle policy (also --throttle-limit=n)<br>- smart: histeresys mechanism on ready tasks<br>- readytasks: throttling based in ready tasks […] |
| -- instrumentation=name | Defines instrumentation plugin:<br>- extrae → creates a Paraver trace<br>- graph → print taskgraph using dot format […] |

- Execution using *4* threads and scheduler *work-first*:

```
$ NX_ARGS="--threads=4 --schedule=wf"  .\cholesky  4096 256
```

# Tracing OmpSs's Applications (paraver)

- Compile and link with instrumentation support

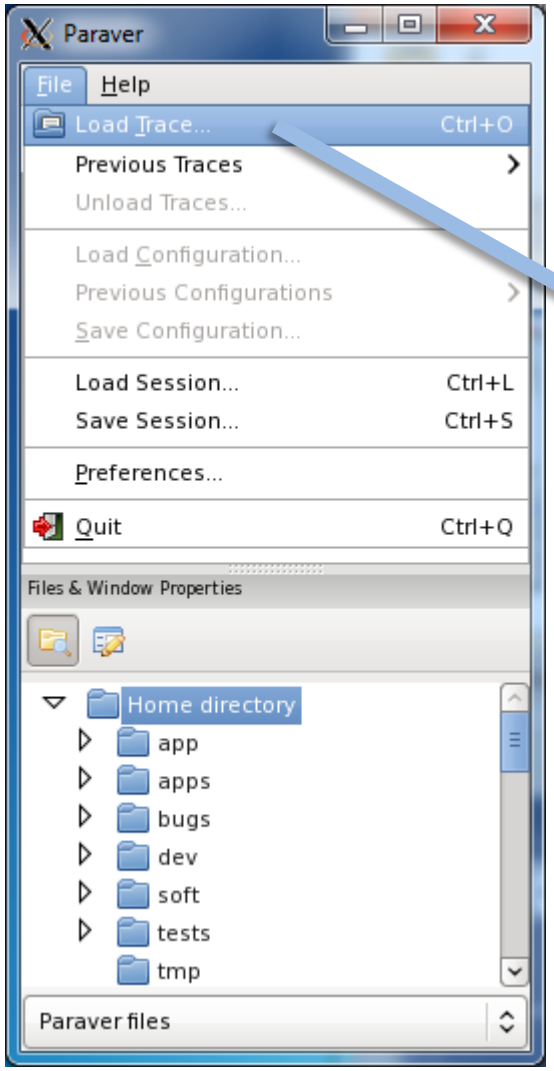
```
$ mcc --ompss --instrument -c cholesky.c
$ mcc -o cholesky --ompss --instrument cholesky.o
```

- When executing specify which instrumentation module

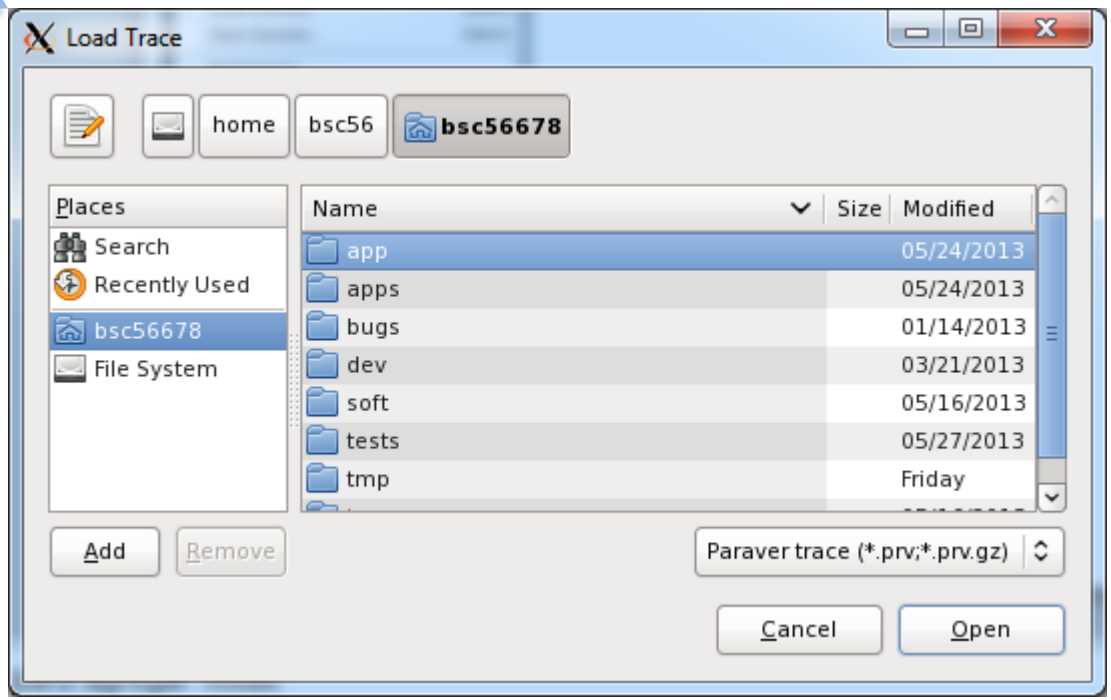
```
NX_INSTRUMENTATION=extrae ./cholesky
```

  - Will generate trace files in executing directory
    - 3 files: .prv, .pcf, .row
    - use *Paraver* to analyze

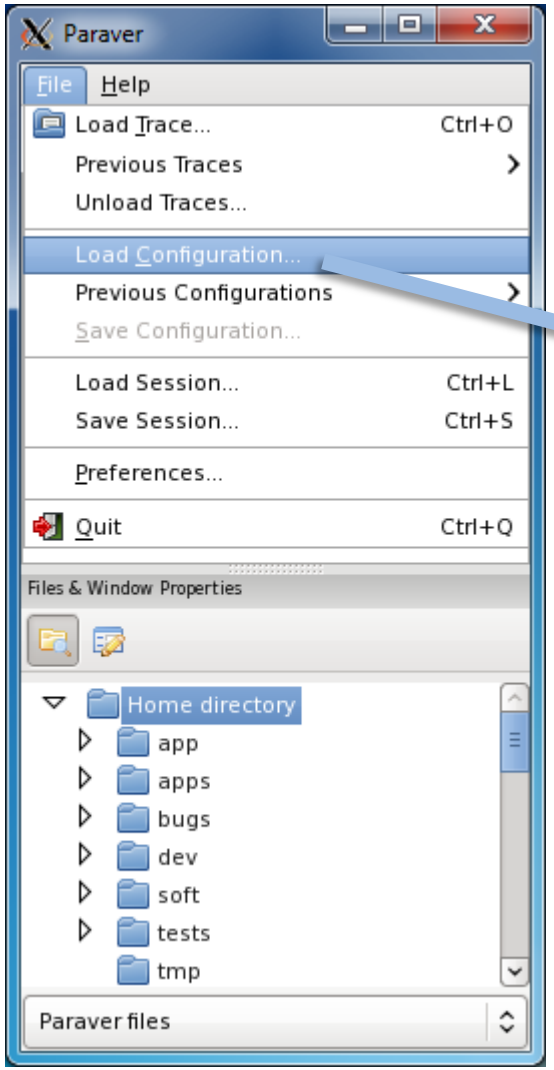
# Paraver Visualization Tool (trace file)



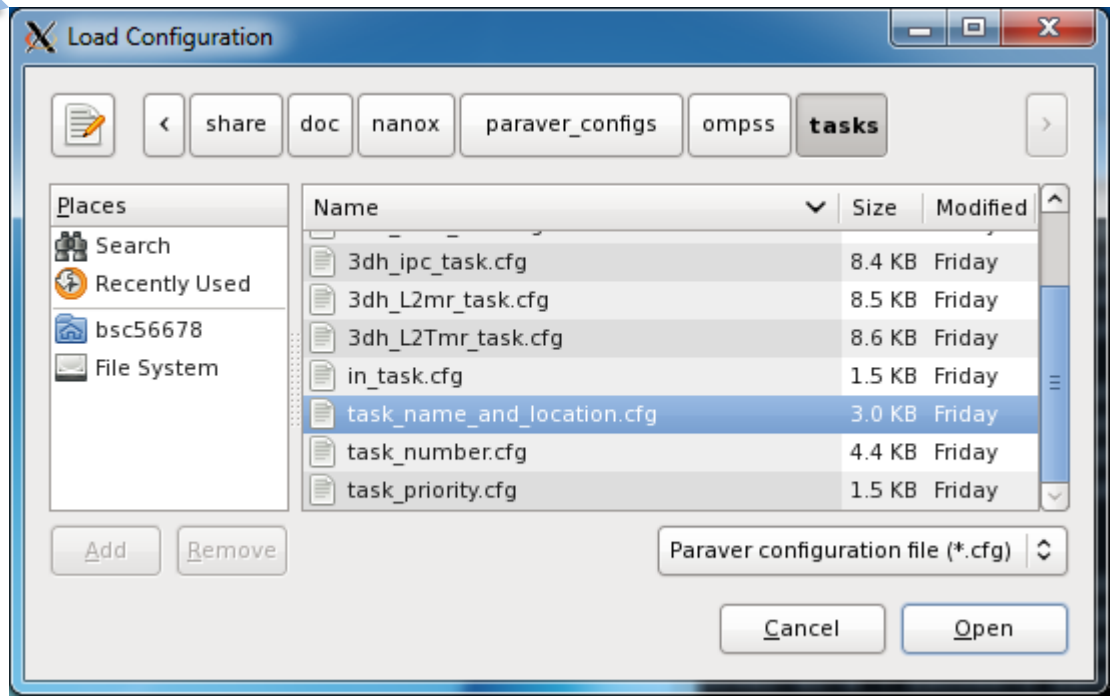

- Using Paraver Visualization Tool
  - Running *wxparaver* command
  - Loading a trace file (.prv)



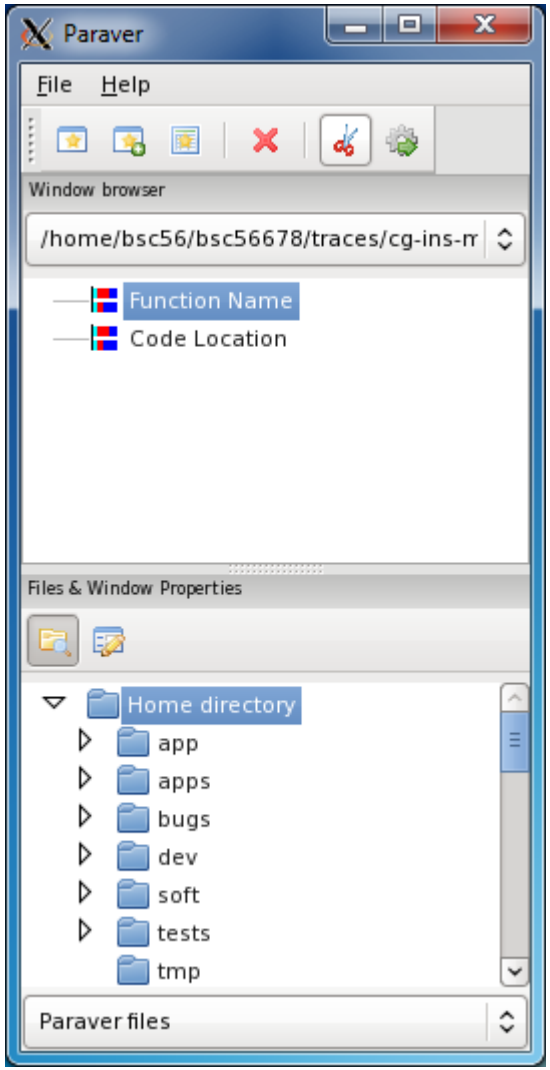
# Paraver Visualization Tool (configuration file) ~ 1



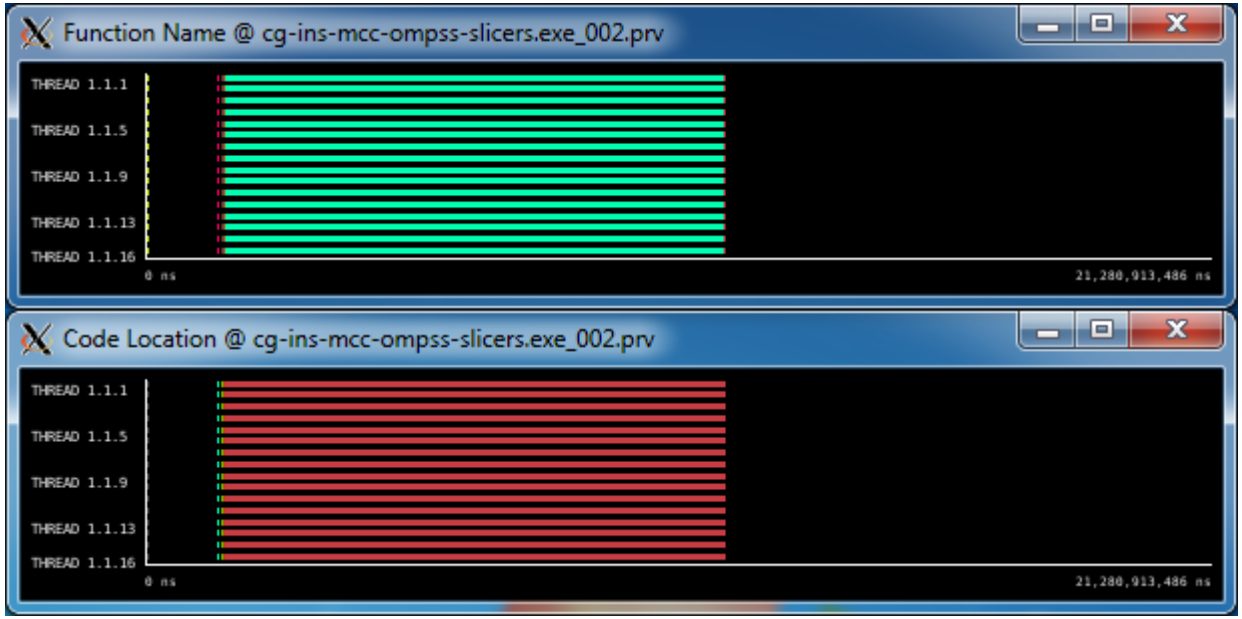

- Using Paraver Visualization Tool
  - Running *wxparaver* command
  - Loading a trace file (.prv)
  - Loading a configuration file (.cfg)



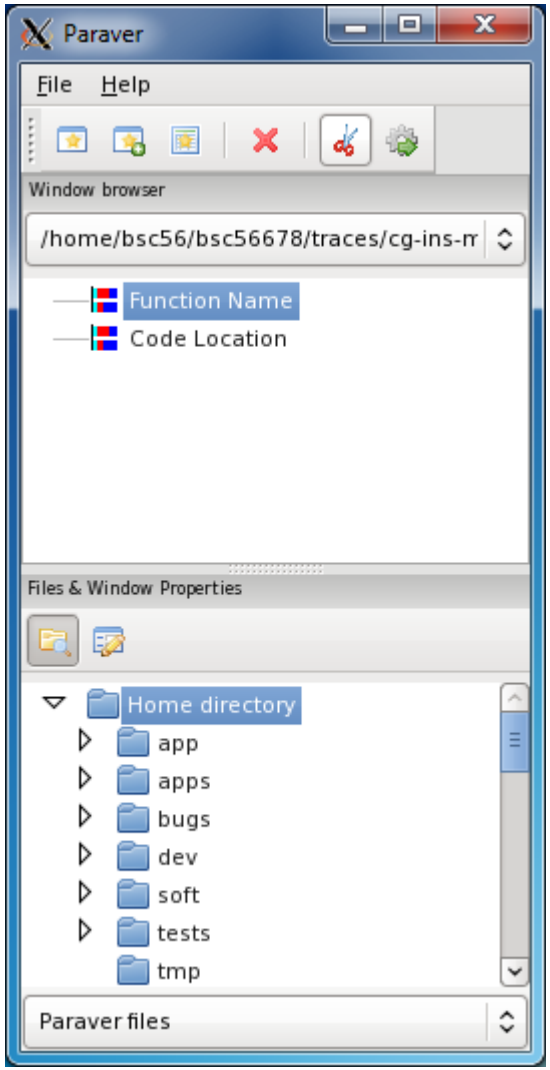
# Paraver Visualization Tool (configuration file) ~ 2

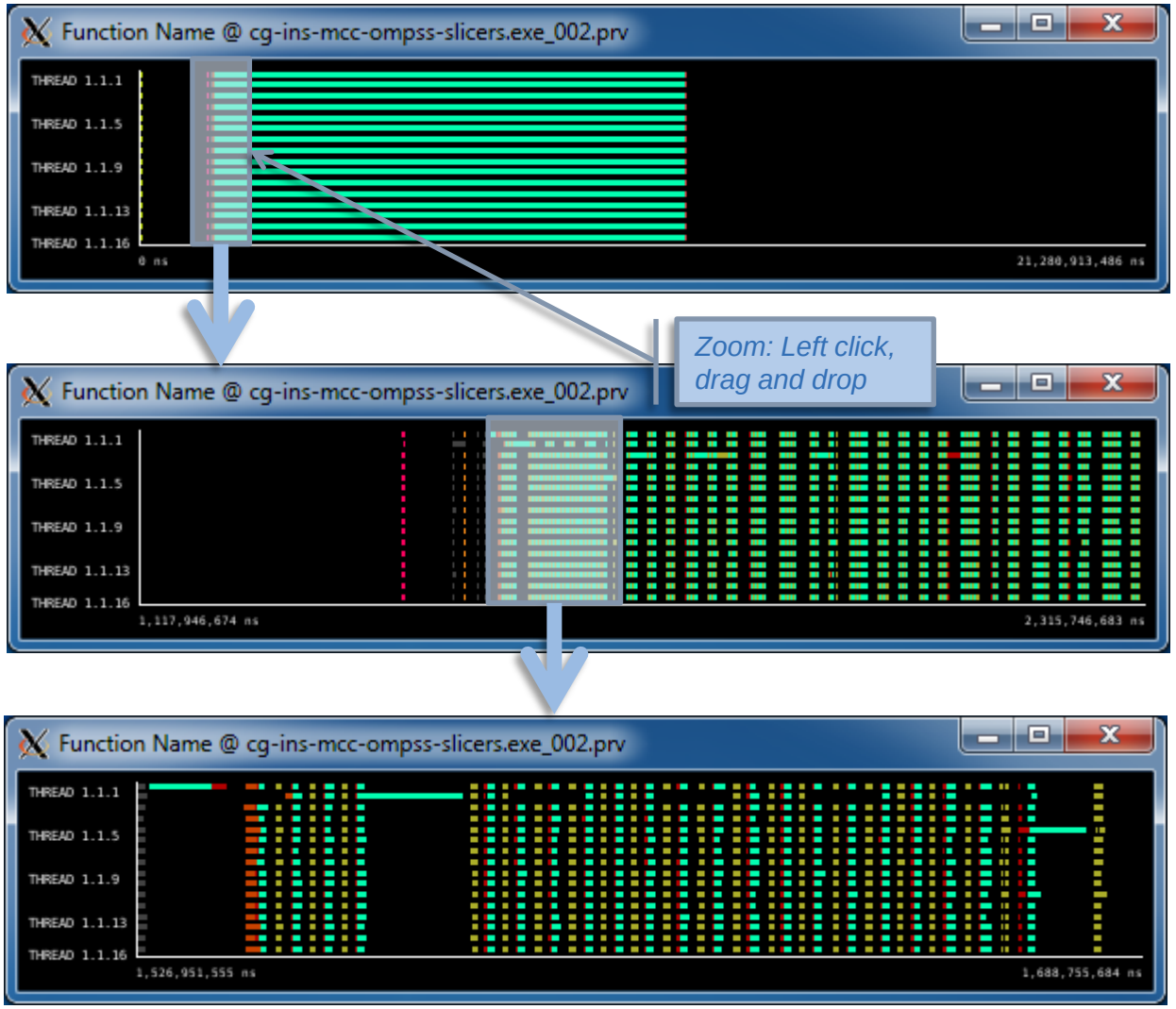



- Using Paraver Visualization Tool
  - Running *wxparaver* command
  - Loading a trace file (.prv)
  - Loading a configuration file (.cfg)

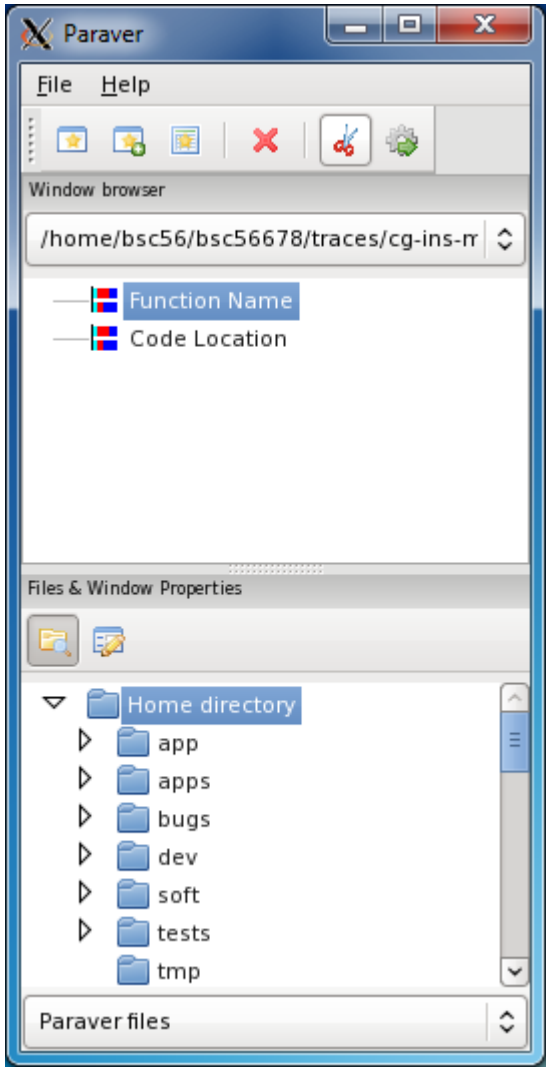


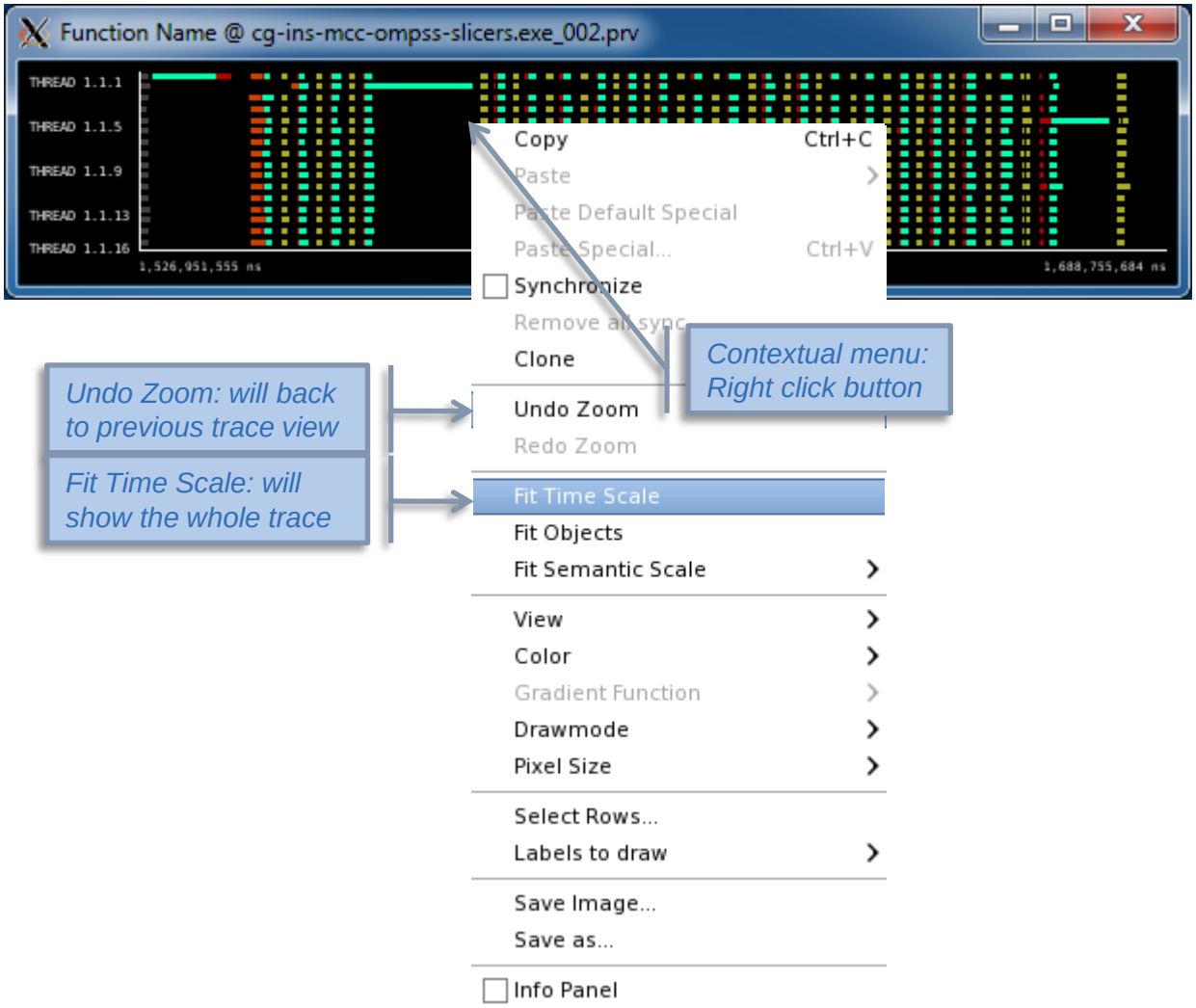
# Paraver Visualization Tool (Zoom In)



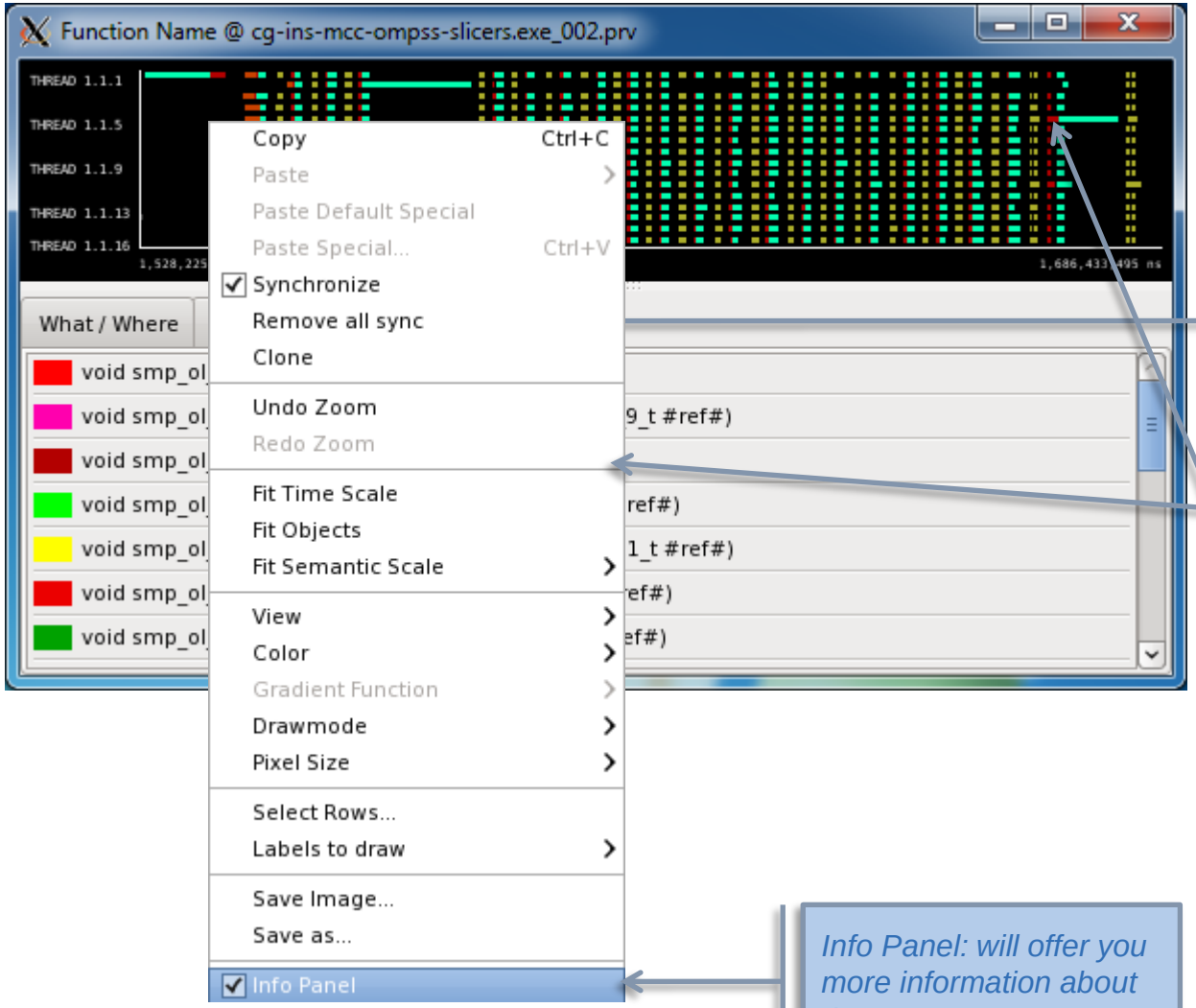



# Paraver Visualization Tool (Zoom Out)



# Paraver Visualization Tool (Info Panel)

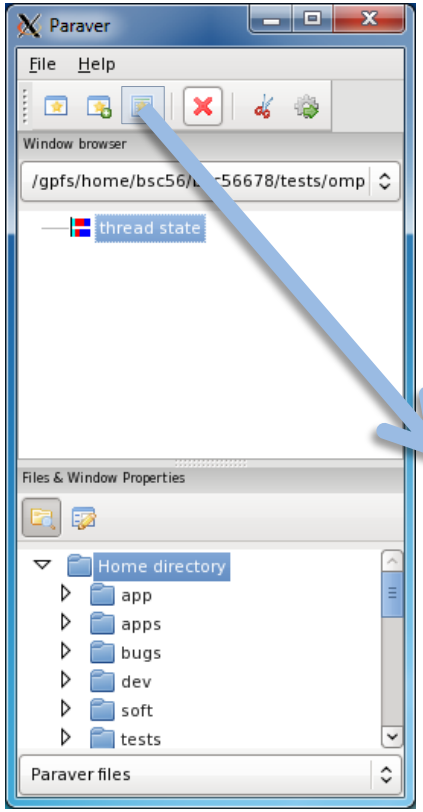


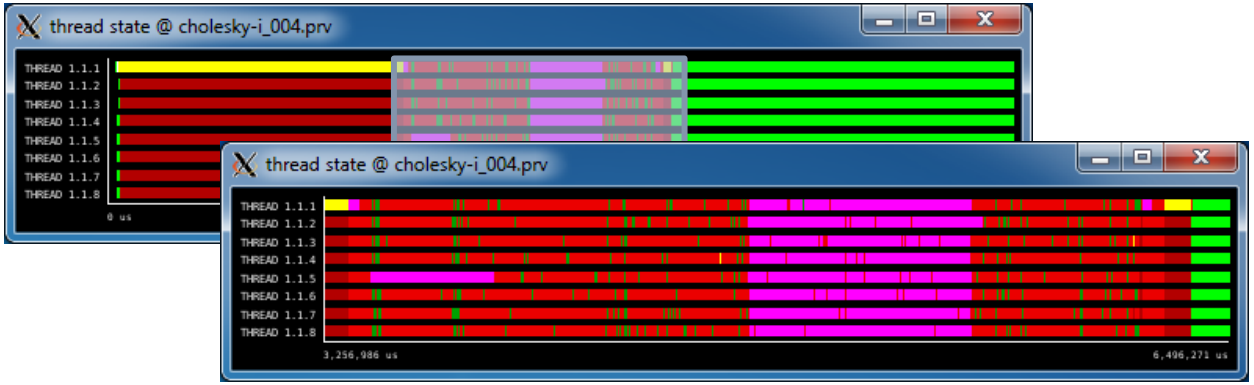

*You can select in between three tabs: what/where, timming and colors*

*Colors Info: relates the semantic value with a given color*

*Info Panel: will offer you more information about the trace…*

# Paraver Visualization Tool (New Histogram) ~ 1




New Histogram #2 @ cholesky-i_004.prv

| | NOT RUNNING | SHUTDOWN | IDLE | RUNTIME | RUNNING | SYNCHRONIZATION | SCHEDULING | CREATION |
|---|---|---|---|---|---|---|---|---|
| THREAD 1.1.1 | 2,127.22 us | 865.85 us | 8,462.90 us | 147,130.69 us | 3,013,496.72 us | 4,400.43 us | 36,248.94 us | 11,831.05 u |
| THREAD 1.1.2 | 1,830.21 us | - | 225,308.93 us | 144,825.46 us | 2,829,984.34 us | 3,885.91 us | 23,020.06 us | 6,031.52 us |
| THREAD 1.1.3 | 6,698.12 us | - | 227,736.02 us | 143,445.03 us | 2,853,036.90 us | 3,670.34 us | 540.60 us | - |
| THREAD 1.1.4 | 1,820.06 us | - | 217,866.03 us | 143,432.70 us | 2,867,264.10 us | 3,762.17 us | 611.74 us | - |
| THREAD 1.1.5 | 1,816.25 us | - | 215,482.19 us | 152,605.95 us | 2,531,799.96 us | 3,640.15 us | 279,704.95 us | 50,172.49 u |
| THREAD 1.1.6 | 1,807.77 us | - | 216,447.14 us | 143,335.89 us | 2,868,907.78 us | 3,775.63 us | 653.02 us | - |
| THREAD 1.1.7 | 1,865.28 us | - | 233,404.83 us | 143,178.92 us | 2,850,748.45 us | 4,229.77 us | 765.35 us | - |
| THREAD 1.1.8 | 1,863.78 us | - | 218,170.30 us | 143,058.39 us | 2,866,644.32 us | 4,107.03 us | 768.11 us | - |
| Total | 19,828.69 us | 865.85 us | 1,562,878.36 us | 1,161,013.03 us | 22,681,882.57 us | 31,471.42 us | 342,312.77 us | 68,035.05 u |
| Average | 2,478.59 us | 865.85 us | 195,359.79 us | 145,126.63 us | 2,835,235.32 us | 3,933.93 us | 42,789.10 us | 22,678.35 u |
| Maximum | 6,698.12 us | 865.85 us | 233,404.83 us | 152,605.95 us | 3,013,496.72 us | 4,400.43 us | 279,704.95 us | 50,172.49 u |
| Minimum | 1,807.77 us | 865.85 us | 8,462.90 us | 143,058.39 us | 2,531,799.96 us | 3,640.15 us | 540.60 us | 6,031.52 us |
| StDev | 1,597.84 us | 0 us | 70,886.78 us | 3,102.87 us | 126,284.14 us | 261.67 us | 90,439.06 us | 19,584.93 u |
| Avg/Max | 0.37 | 1 | 0.84 | 0.95 | 0.94 | 0.89 | 0.15 | 0.45 |

*Creating New Histogram using Current Timeline*

# Tracing OmpSs's Applications (graph)

- Compile and link with instrumentation support

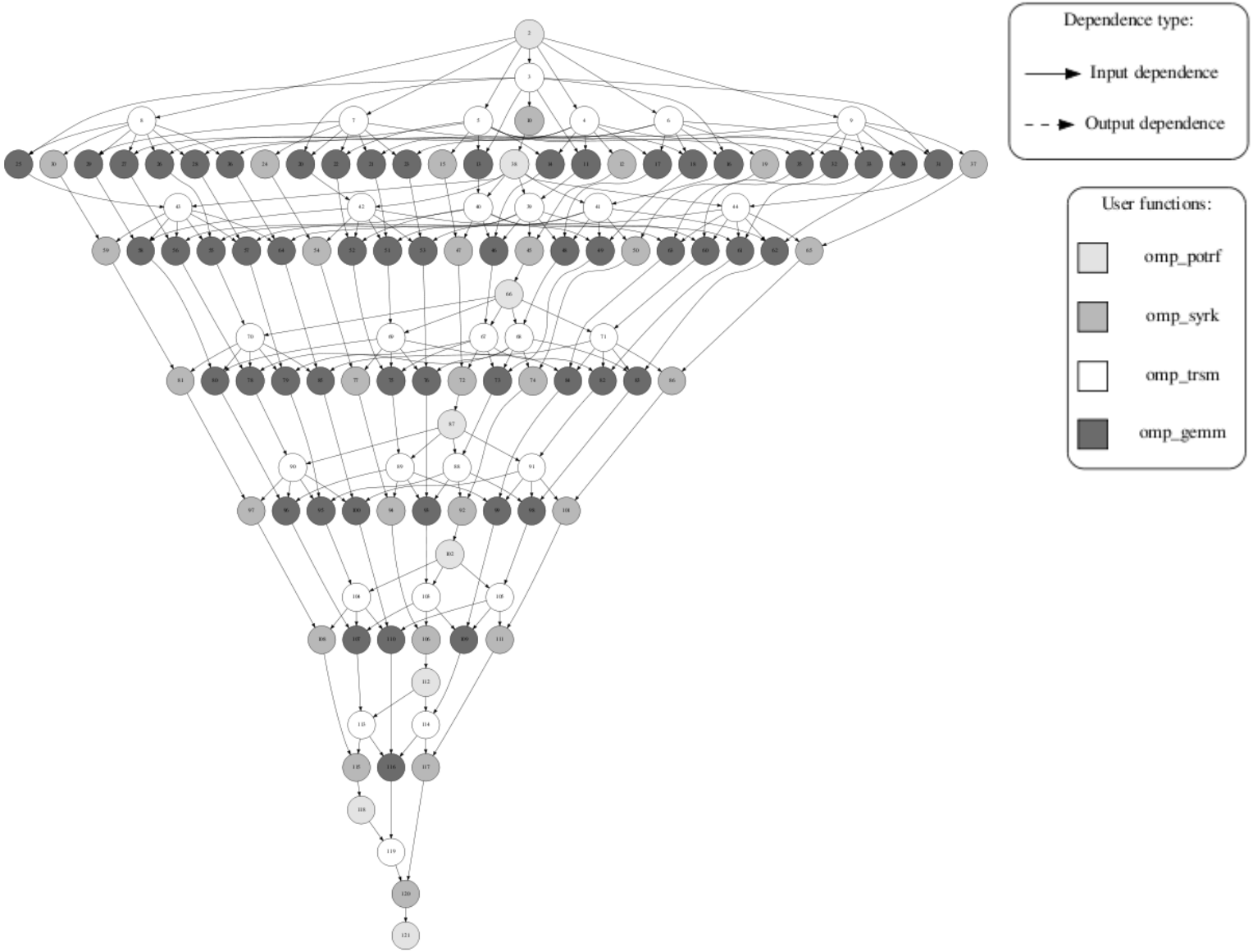
```
$ mcc --ompss --instrument -c cholesky.c
$ mcc -o cholesky --ompss --instrument cholesky.o
```

- When executing specify which instrumentation module

```
NX_THREADS=1  NX_INSTRUMENTATION=graph ./cholesky
```

  - Will generate a dependence graph in .dot and .pdf formats

# Graph Instrumentation: Cholesky 4096x4096 – 512x512



# Hands-on Exercises: Machine Description

- Minotauro (system overview)
  - 2 Intel E5649 6C at 2.53 GHz
  - 2 M2090 NVIDIA GPU Cards
  - 24 GB of Main memory
  - Peak Performance: 185.78 TFlops
  - 250 GB SSD as local storage
  - 2 Infiniband QDR (40 Gbit each)
- Top 500 Ranking (11/2012):

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11/2012 | 290 | Bullx B505, Xeon E5649 6C 2.53GHz, Infiniband QDR, NVIDIA 2090 | Bull SA | 5544 | 103.2 | 182.9 | 81.50 |
| 06/2012 | 177 | Bullx B505, Xeon E5649 6C 2.53GHz, Infiniband QDR, NVIDIA 2090 | Bull SA | 5544 | 103.2 | 182.9 | 81.50 |
| 11/2011 | 114 | Bullx B505, Xeon E5649 6C 2.53GHz, Infiniband QDR, NVIDIA 2090 | Bull SA | 5544 | 103.2 | 182.9 | 81.50 |

# Hands-on Exercises: Methodology

- Exercises in:
  - ompss-exercises-MT.tar.gz → **01-compile_and_run**
    - cholesky
    - stream_dep
    - stream_barr
  - ~~# @ partition = debug~~ →
  - # @ reservation = summer-school (*at job script:* **run-x.sh**)
- Paraver configuration files → Organized in directories
  - **tasks**: task related information
  - **runtime**: runtime internals events
  - **scheduling**: task-graph and scheduling
  - **cuda**: specific to CUDA implementations (GPU devices)
  - **data-transfer**: specific for data send and receive operations
  - **mpi-ompss**: specific to MPI + OmpSs hybrid parallelization

www.bsc.es


**Barcelona Supercomputing Center**
*Centro Nacional de Supercomputación*

# Thank you!

For further information please contact
pm-tools@bsc.es

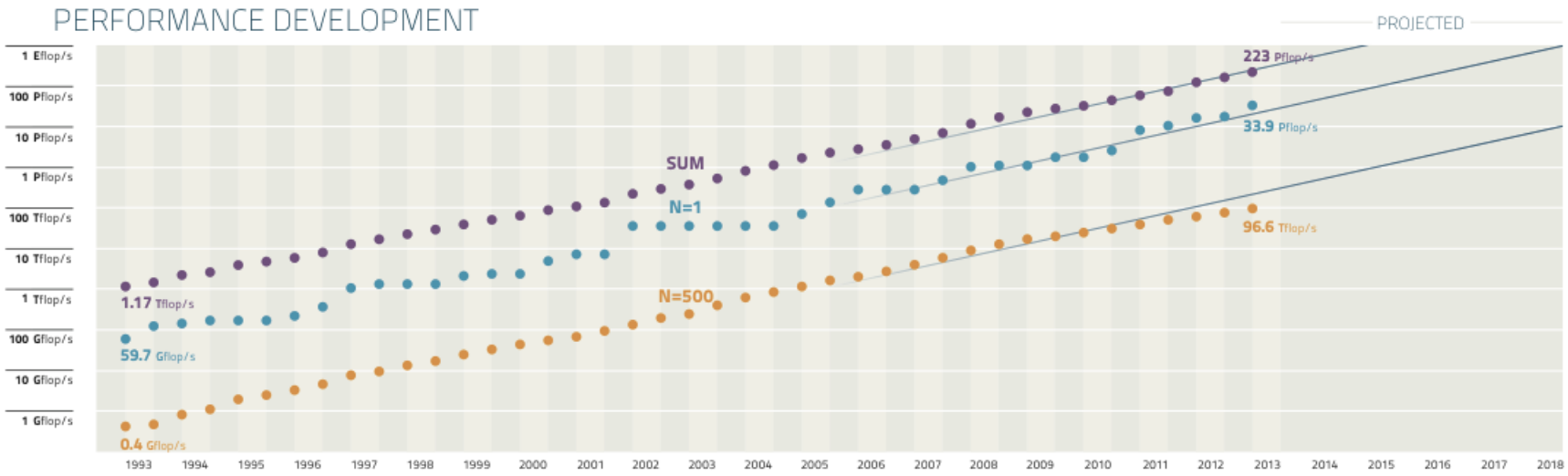
# TOP 500 List: Performance development

TOP500® JUNE 2013

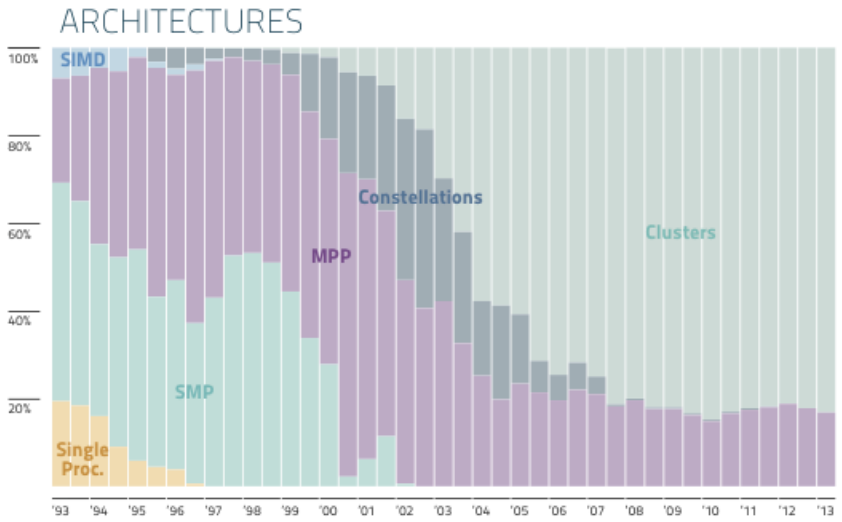
PRESENTED BY UNIVERSITY OF MANNHEIM · ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE · BERKELEY LAB Lawrence Berkeley National Laboratory

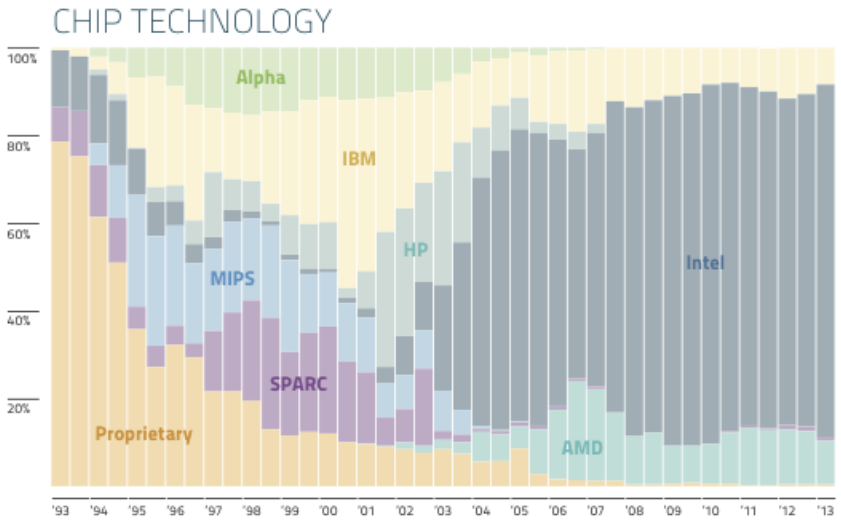
FIND OUT MORE AT www.top500.org

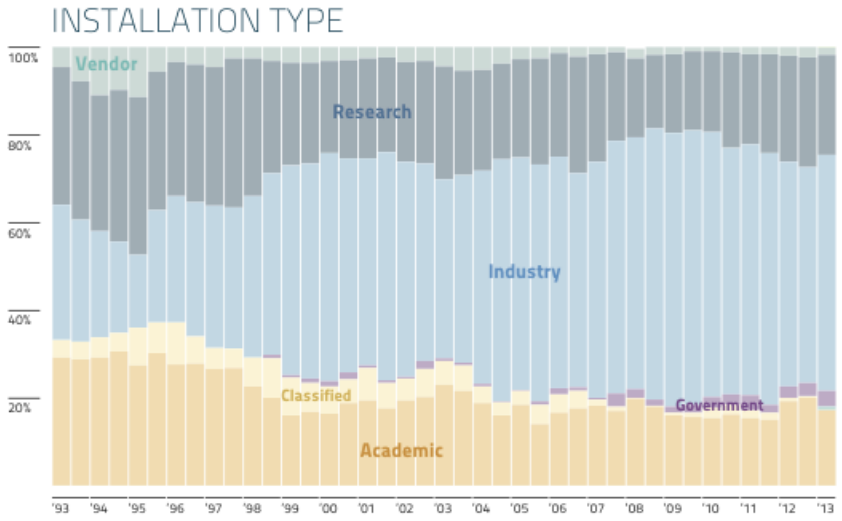
| | NAME | SPECS | SITE | COUNTRY | CORES | $R_{MAX}$ PFLOP/S | POWER MW |
|---|---|---|---|---|---|---|---|
| 1 | **Tianhe-2 (Milkyway-2)** | NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect | NUDT | China | 3,120,000 | **33.9** | 17.8 |
| 2 | **Titan** | Cray XK7, Opteron 6274 (16C, 2.2 GHz) + Nvidia Kepler (14C, .732 GHz), Custom interconnect | DOE/SC/ORNL | USA | 560,640 | **17.6** | 8.3 |
| 3 | **Sequoia** | IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect | DOE/NNSA/LLNL | USA | 1,572,864 | **17.2** | 7.9 |
| 4 | **K computer** | Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect | RIKEN AICS | Japan | 705,024 | **10.5** | 12.7 |
| 5 | **Mira** | IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect | DOE/SC/ANL | USA | 786,432 | **8.16** | 3.95 |

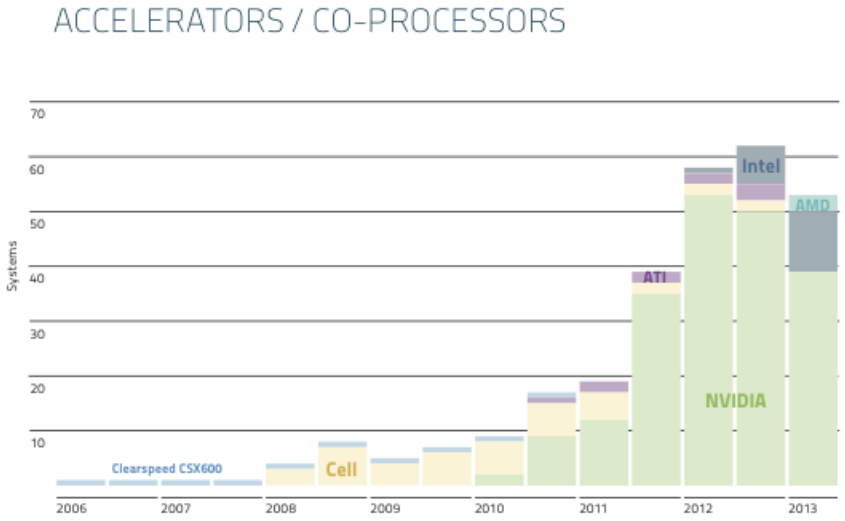


# TOP 500 List: Statistics



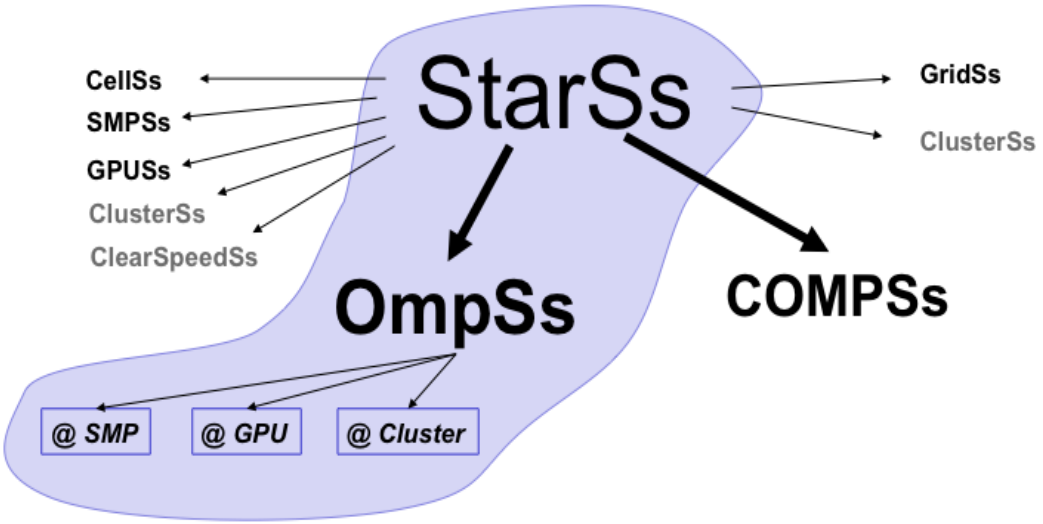



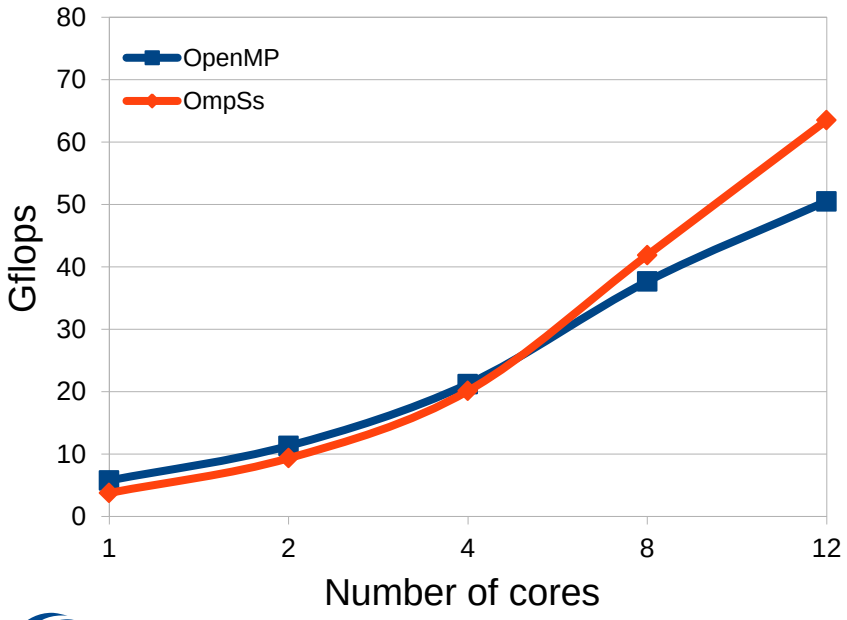
# OmpSs Introduction ~ 2

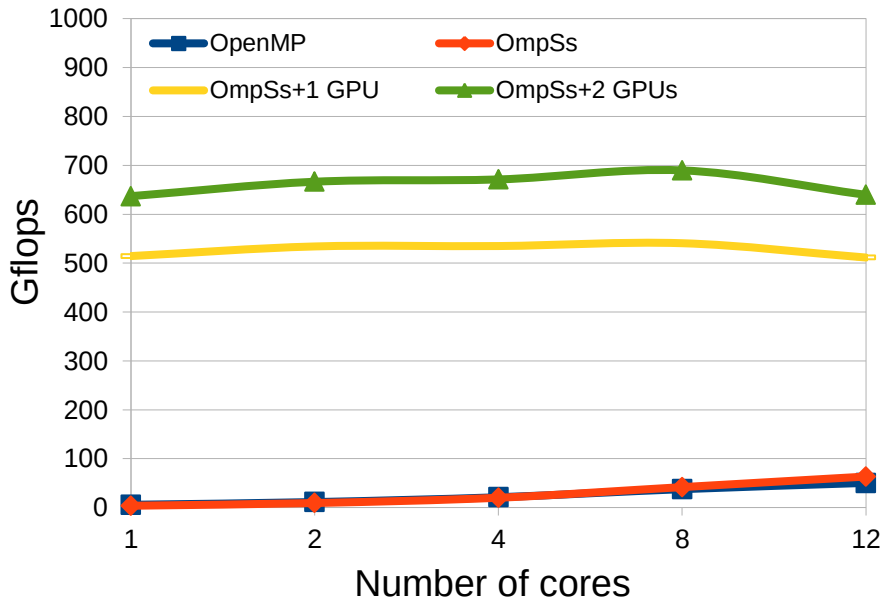
- StarSs: a family of task based programming models
  - Basic concept: write sequential on a flat single address space + directionality annotations
  - Order IS defined
  - Dependence and data access related information (NOT specification) in a single mechanism
  - Think global, specify local
  - Intelligent Runtime



# Cholesky Factorization (performance results - CPUs)

- Cholesky parameters
  - Matrix (total) size → 16 K
  - Block/Tile Size → 256 (*single float's*)
- System Overview *(per node)* → *Minotauro*
  - **2 Intel E5649 6C at 2.53 GHz +** 2 M2090 NVIDIA GPU Cards
  - 24 GB of Main memory + 250 GB SSD as local storage



# Executing OmpSs Applications (Nanos++ Options) ~ 3

- Nanos++ utility: getting help using command line
  - Listing available modules

```
$ nanox --list-modules
```

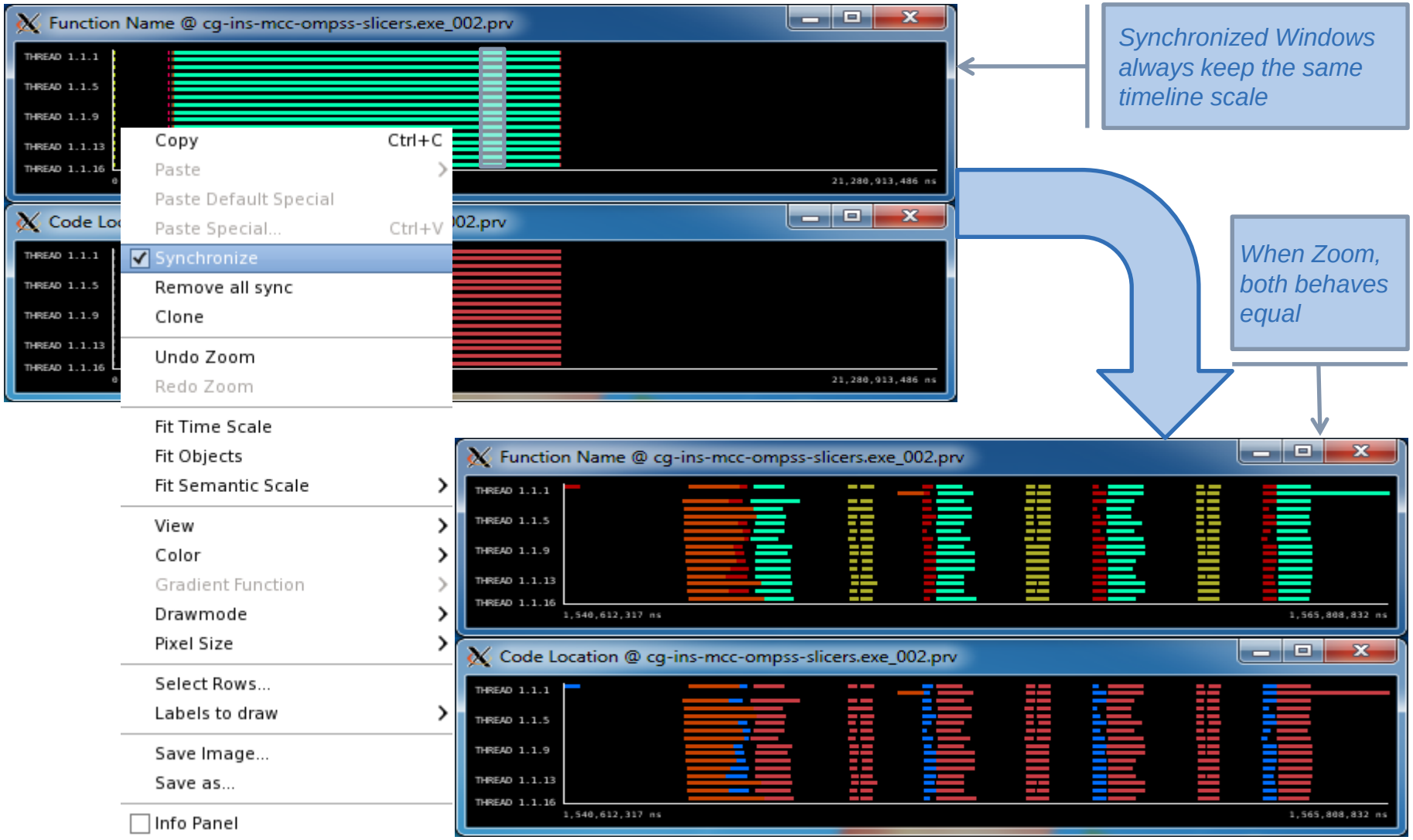
  - Listing available options

```
$ nanox –help
```

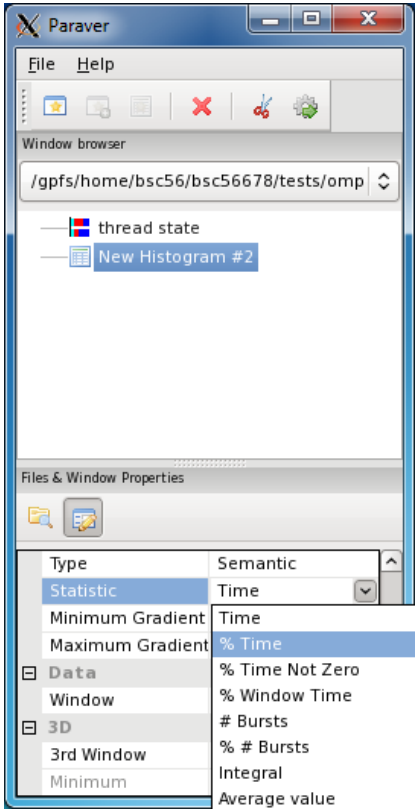
  - Getting installed version (*reporting a bug*)

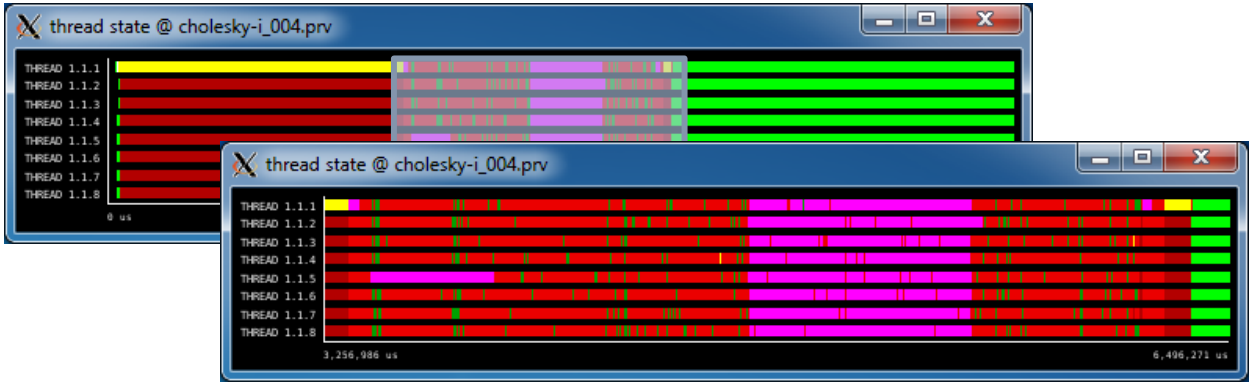
```
$ nanox --version
nanox 0.7a (git master c1f548a 2013-05-27 17:55:54 +0200 dev)
Configured with: configure --prefix=/usr/bin --disable-debug
--enable-gpu-arch --with-cuda=/opt/cuda/4.1/
```

# Paraver Visualization Tool (Synchronized Windows)



# Paraver Visualization Tool (New Histogram) ~ 2




New Histogram #2 @ cholesky-i_004.prv

| | NOT RUNNING | SHUTDOWN | IDLE | RUNTIME | RUNNING | SYNCHRONIZATION | SCHEDULING | CREATION |
|---|---|---|---|---|---|---|---|---|
| THREAD 1.1.1 | 0.07 % | 0.03 % | 0.26 % | 4.56 % | 93.45 % | 0.14 % | 1.12 % | 0.37 % |
| THREAD 1.1.2 | 0.06 % | - | 6.96 % | 4.48 % | 87.48 % | 0.12 % | 0.71 % | 0.19 % |
| THREAD 1.1.3 | 0.21 % | - | 7.04 % | 4.43 % | 88.19 % | 0.11 % | 0.02 % | - |
| THREAD 1.1.4 | 0.06 % | - | 6.74 % | 4.43 % | 88.64 % | 0.12 % | 0.02 % | - |
| THREAD 1.1.5 | 0.06 % | - | 6.66 % | 4.72 % | 78.26 % | 0.11 % | 8.65 % | 1.55 % |
| THREAD 1.1.6 | 0.06 % | - | 6.69 % | 4.43 % | 88.69 % | 0.12 % | 0.02 % | - |
| THREAD 1.1.7 | 0.06 % | - | 7.22 % | 4.43 % | 88.14 % | 0.13 % | 0.02 % | - |
| THREAD 1.1.8 | 0.06 % | - | 6.74 % | 4.42 % | 88.62 % | 0.13 % | 0.02 % | - |
| Total | 0.61 % | 0.03 % | 48.32 % | 35.91 % | 701.48 % | 0.97 % | 10.58 % | 2.10 % |
| Average | 0.08 % | 0.03 % | 6.04 % | 4.49 % | 87.68 % | 0.12 % | 1.32 % | 0.70 % |
| Maximum | 0.21 % | 0.03 % | 7.22 % | 4.72 % | 93.45 % | 0.14 % | 8.65 % | 1.55 % |
| Minimum | 0.06 % | 0.03 % | 0.26 % | 4.42 % | 78.26 % | 0.11 % | 0.02 % | 0.19 % |
| StDev | 0.05 % | 0 % | 2.19 % | 0.10 % | 3.96 % | 0.01 % | 2.80 % | 0.61 % |
| Avg/Max | 0.37 | 1 | 0.84 | 0.95 | 0.94 | 0.89 | 0.15 | 0.45 |

*- Windows Properties*
*- Statistics*
*- Statistic → (%)*