www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tutorial OmpSs: Development methodology and infrastructure

PUMPS 2013 tutorial
Hybrid and Heterogeneous Parallel Programming with MPI/ OmpSs for Exascale Syste

Rosa M Badia, Xavier Martorell

EXCELENCIA SEVERO OCHOA

PUMPS 2013, 12 July 2013

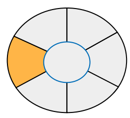
Tutorial OmpSs

- Agenda

9:00 – 10:30	Introduction to StarSs OmpSs syntax Simple examples Development methodology and infrastructure	90 min
10:30 – 11:00	Coffee break	30 min
11:10 – 12:30	Hands-on single node (I)	90 min
12:30 – 13:30	Lunch	60 min
13:30 – 15:00	Support for heterogeneous platforms Advanced examples	90 min
15:00 – 15:15	Short break	15 min
15:15 – 17:00	Hands-on (II)	105 min

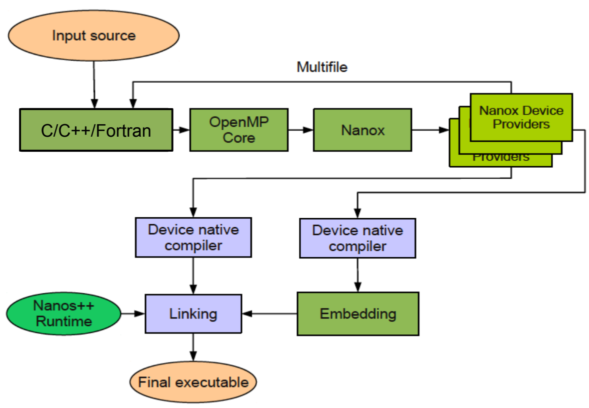
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

OmpSs compiler and runtime



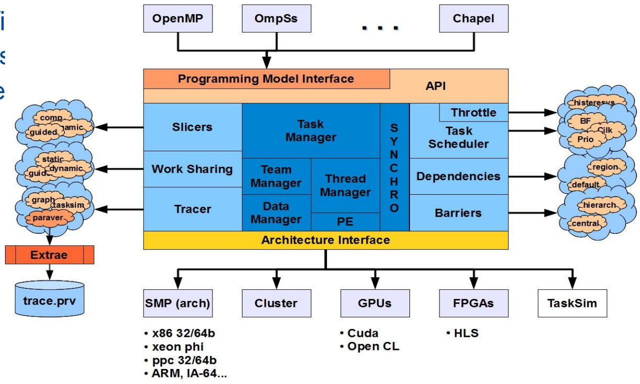
Mercurium Compiler

- Recognizes constructs and transforms them to calls to the runtime
- Manages code restructuring for different target devices
  - Device-specific handlers
  - May generate code in a separate file
  - Invokes different back-end compilers
    → nvcc for NVIDIA



The NANOS++ Runtime

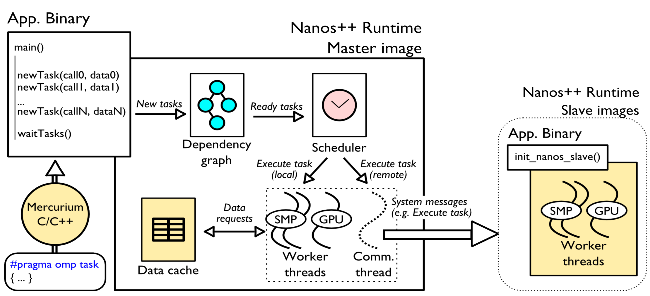
- Nanos++
  - Common execution runtime (C, C++ and Fortran)
  - Target specific features
  - Task creation, dependency management, resilience, …
  - Task scheduling (BF, Cilk, Priority, Socket, …)
  - Data management: Unifi
    - Transparently manages
    - … and data transfer be



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime structure behaviour: task handling

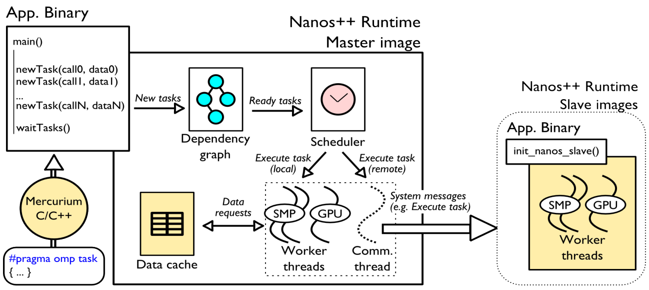
- Task generation
- Data dependence analysis
- Task scheduling



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime structure behaviour: coherence support

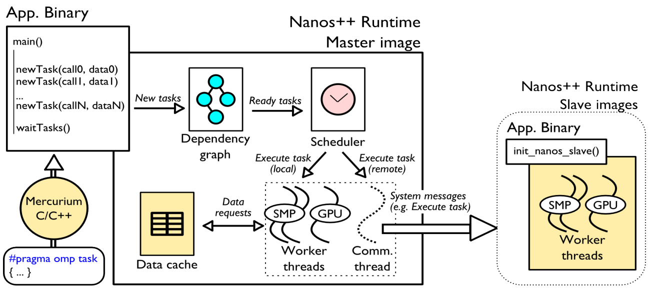
- Different address spaces managed with:
  - A hierarchical directory
  - A software cache per each:
    - Cluster node
    - GPU
- Data transfers between different memory spaces only when needed
  - Write-through
  - Write-back



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime structure behaviour: GPUs

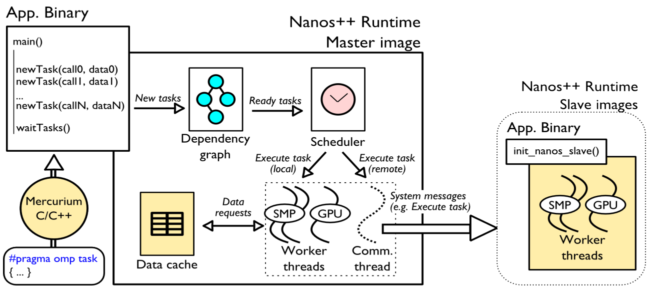
- Automatic handling of Multi-GPU execution
- Transparent data-management on GPU side (allocation, transfers, ...) and synchronization
- One manager thread in the host per GPU. Responsible for:
  - Transferring data from/to GPUs
  - Executing GPU tasks
  - Synchronization
- Overlap of computation and communication
- Data pre-fetch



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime structure behaviour: clusters

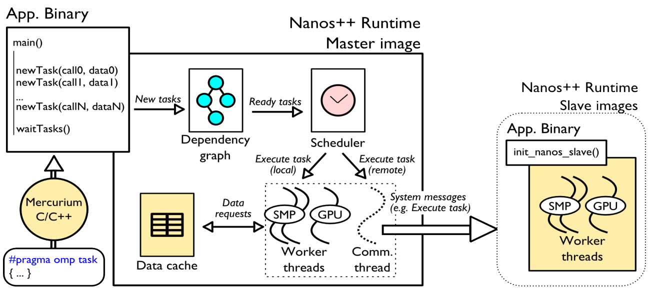
- One runtime instance per node
  - One master image
  - N-1 slave images
- Low level communication through active messages
- Tasks generated by master
  - Tasks executed by worker threads in the master
  - Tasks delegated to slave nodes through the communication thread
- Remote task execution:
  - Data transfer (if necessary)
  - Overlap of computation with communication
  - Task execution
    - Local scheduler



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Runtime structure behavior: clusters of GPUs

- Composes previous approaches
- Supports for heterogeneity and hierarchy:
  - Application with homogeneous tasks: SMP or GPU
  - Applications with heterogeneous tasks: SMP and GPU
  - Applications with hierarchical and heterogeneous tasks:
    - I.e., coarser grain SMP tasks
    - Internally generating GPU tasks



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Compiling

- Compiling

```
mcc --ompss  -c bin.c
```

- Linking

```
mcc --ompss -o bin bin.o
```

- where frontend can be:

mcc	C
mcxx	C++
mnvcc	CUDA & C
mnvcxx	CUDA & C++
mfc	Fortran

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Compiling

- Compatibility flags:
  - -I, -g, -L, -l, -E, -D, -W
- Other compilation flags:

-k	Keep intermediate files
--debug	Use Nanos++ debug version
--instrumentation	Use Nanos++ instrumentation version
--version	Show Mercurium version number
--verbose	Enable Mercurium verbose output
--Wp,flags	Pass flags to preprocessor (comma separated)
--Wn,flags	Pass flags to native compiler (comma separated)
--Wl,flags	Pass flags to linker (comma separated)
--help	To see many more options :-)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Executing

- No LD_LIBRARY_PATH or LD_PRELOAD needed

```
./bin
```

- Adjust number of threads with OMP_NUM_THREADS

```
OMP_NUM_THREADS=4 ./bin
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Nanos++ options

- Other options can be passed to the Nanos++ runtime via NX_ARGS

```
NX_ARGS="options" ./bin
```

--schedule=name	Use name task scheduler
--throttle=name	Use name throttle-policy
--throttle-limit=limit	Limit of the throttle-policy (exact meaning depends on the policy)
--instrumentation=name	Use name instrumentation module
--disable-yield	Nanos++ won't yield threads when idle
--spins=number	Number of spin loops when idle
--disable-binding	Nanos++ won't bind threads to CPUs
--binding-start=cpu	First CPU where a thread will be bound
--binding-stride=number	Stride between bound CPUs

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Nanox helper

- Nanos++ utility to
  - list available modules:

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

  - list available options:

```
nanox --help
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tracing

- Compile and link with --instrument

```
mcc --ompss --instrument -c bin.c
mcc -o bin --ompss --instrument bin.o
```

- When executing specify which instrumentation module to use:

```
NX_INSTRUMENTATION=extrae ./bin
```

- Will generate trace files in executing directory
  - 3 files: prv, pcf, rows
  - Use paraver to analyze

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Reporting problems

- Compiler problems
  - http://pm.bsc.es/projects/mcxx/newticket
- Runtime problems
  - http://pm.bsc.es/projects/nanox/newticket
- Support mail
  - pm-tools@bsc.es
- Please include snapshot of the problem

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming methodology

- Correct sequential program
- Finding tasks with Tareador
- Debugging with Ayudame/Temanejo
- Incremental taskification
  - Test every individual task with forced sequential in-order execution
    - → 1 thread, scheduler = FIFO, throtle=1
- Single thread out-of-order execution
- Increment number of threads
  - Use taskwaits to force certain levels of serialization

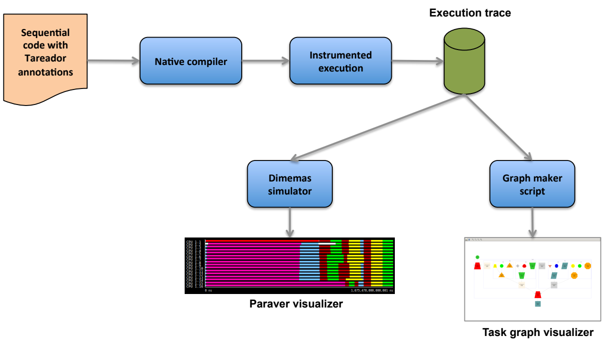
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Finding tasks with Tareador

- From a sequential specification of the program ...
  - ...Find a decomposition of the problem in tasks (i.e. Identify pieces of work that can execute concurrently) ...
  - ... ensuring that the same result is produced (i.e. Identify dependencies that impose ordering and data sharing constraints).

Guiding the task decomposition process

- Tareador: environment to support the task decomposition process:
  - API to annotate sequential program with potential tasks
  - Binary instrumentation using a new Valgrind module
  - Visualization of task graph (granularities and dependences)
  - Simulation with Dimemas and visualization with Paraver



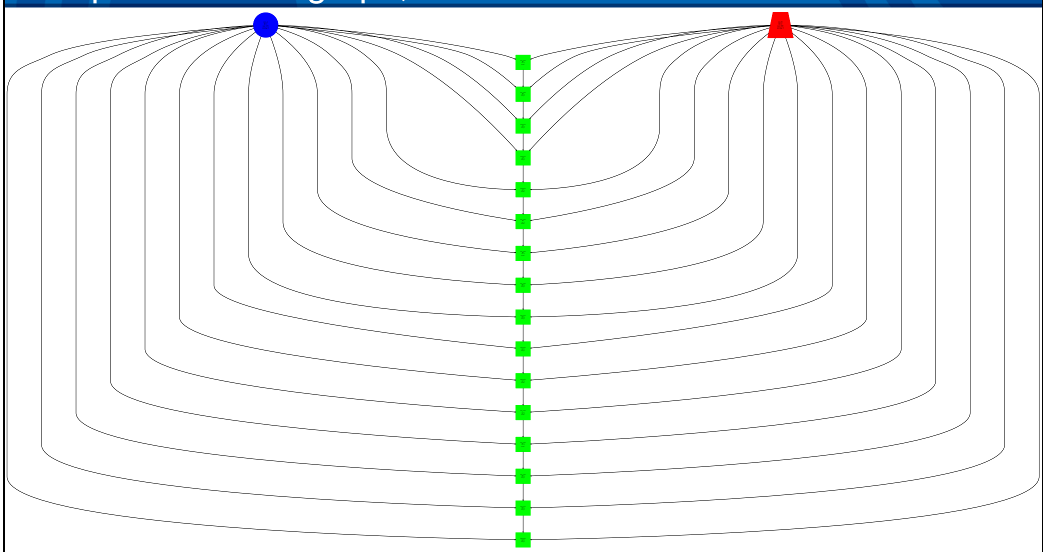
Example code

```
void dot_product (long N,
    double A[N], double B[N], double *acc){
    double prod;

    *acc=0.0;
    for (int i=0; i<N; i++) {
        tareador_start_task("inner_product");
        prod = A[i]*B[i];
        *acc+= prod;
        tareador_end_task();
    }
}
```

```
int main() {
    tareador_ON ();

    tareador_start_task("init A");
    for (int i=0; i< N; i++) A[i]=i;
    tareador_end_task();

    tareador_start_task("init B");
    for (int i=0; i< N; i++) B[i]=2*i;
    tareador_end_task();

    dot_product (N, A, B, &result);

    tareador_OFF ();
}
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

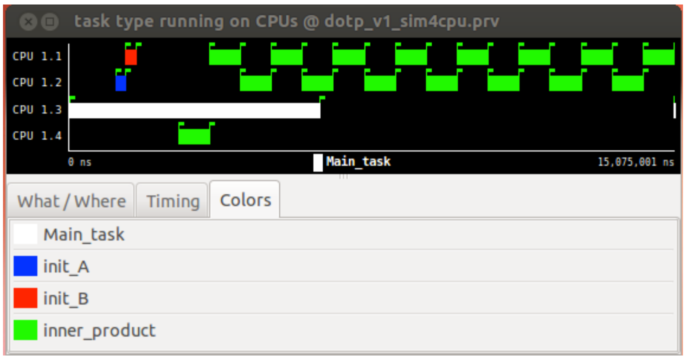
Dot product: task graph, N=16



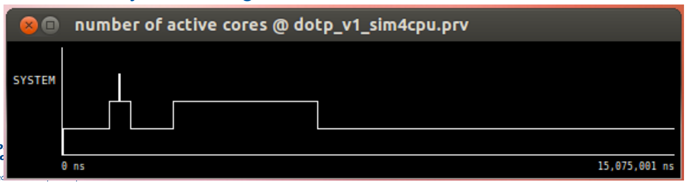
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dot product: Dimemas simulation, N=16, 4 CPU

CPU view load configuration



Maximum concurrency view conguration



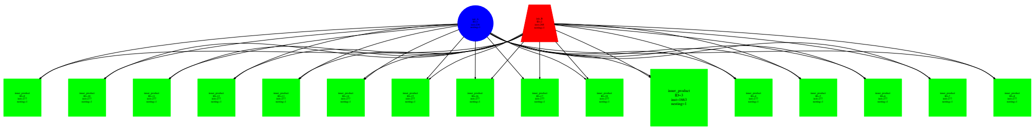
Example code

```
void dot_product (long N,
    double A[N], double B[N], double *acc){
    double prod;

    *acc=0.0;
    for (int i=0; i<N; i++) {
        tareador_start_task("inner_product");
        prod = A[i]*B[i];
        tareador_disable_object(acc);
        *acc+= prod;
        tareador_enable_object(acc);
        tareador_end_task();
    }
}
```

```
int main() {
    tareador_ON ();

    tareador_start_task("init A");
    for (int i=0; i< N; i++) A[i]=i;
    tareador_end_task();

    tareador_start_task("init B");
    for (int i=0; i< N; i++) B[i]=2*i;
    tareador_end_task();

    dot_product (N, A, B, &result);

    tareador_OFF ();
}
```

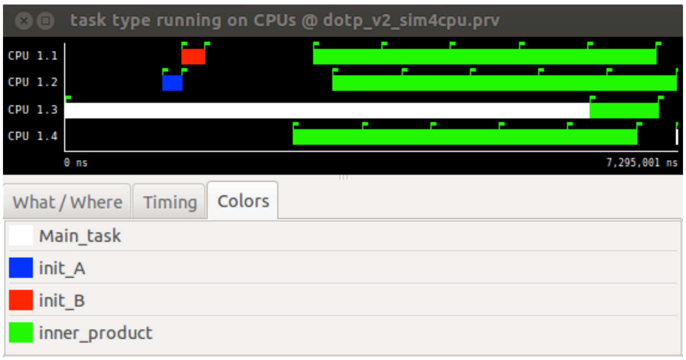
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dot product: task graph, N=16

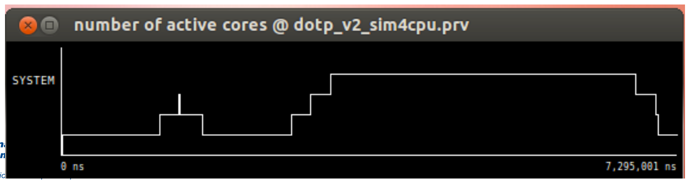


Dot product: Dimemas simulation, N=16, 4 CPU

CPU view load configuration

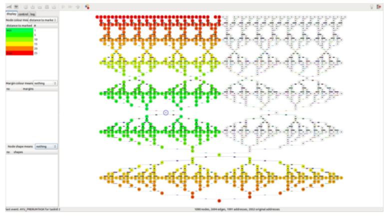


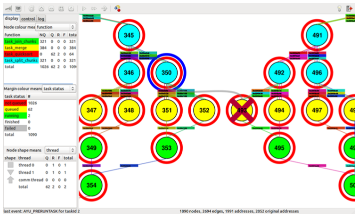
Maximum concurrency view configuration



Debugging: AYUDAME/TEMANEJO

- Leverage probe hooks provided by compiler and runtime
- Task based debugging:
  - Display graph
  - Control execution environment (#threads,…)
  - Breakpoints at tasks
- Interface to instruction level debugger (gdb)





By HLRS

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

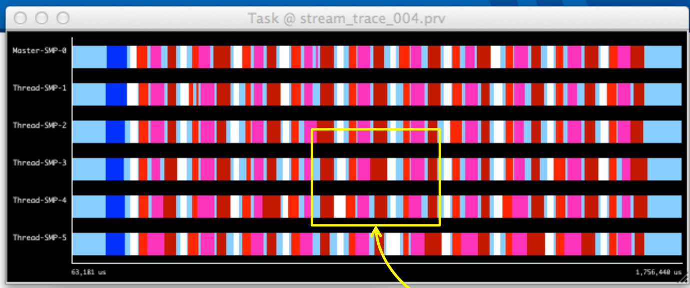
Visualizing Paraver tracefiles

- Set of Paraver configuration files ready for OmpSs. Organized in directories
  - **Tasks: related to application tasks**
  - Runtime, nanox-configs: related to OmpSs runtime internals
  - **Graph_and_scheduling: related to task-graph and task scheduling**
  - DataMgmgt: related to data management
  - CUDA: specific to GPU

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

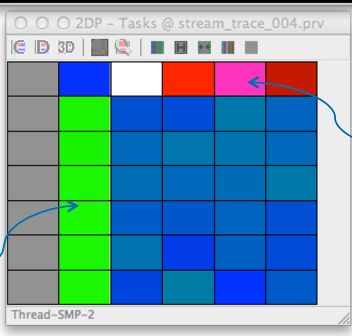
Tasks’ profile

- 2dp_tasks.cfg
- Tasks’ profile



control window:
timeline where each
color represent the
task been executed
by each thread

light blue: not executing
tasks



threads

different colours
represent different
task type

gradient color,
indicates given estadístic:
i.e., number of tasks instances

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

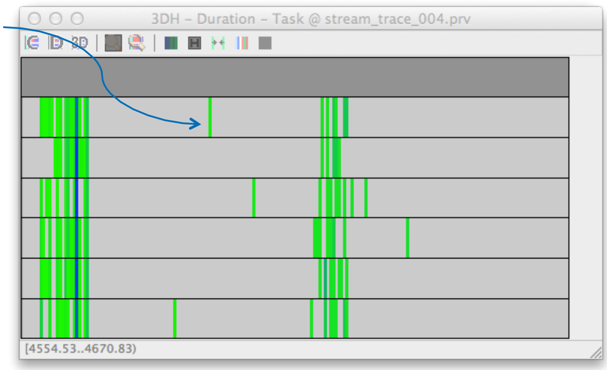
Tasks duration histogram

- 3dh_duration_task.cfg

time intervals

gradient color,
indicates given estadístic:
i.e., number of tasks instances

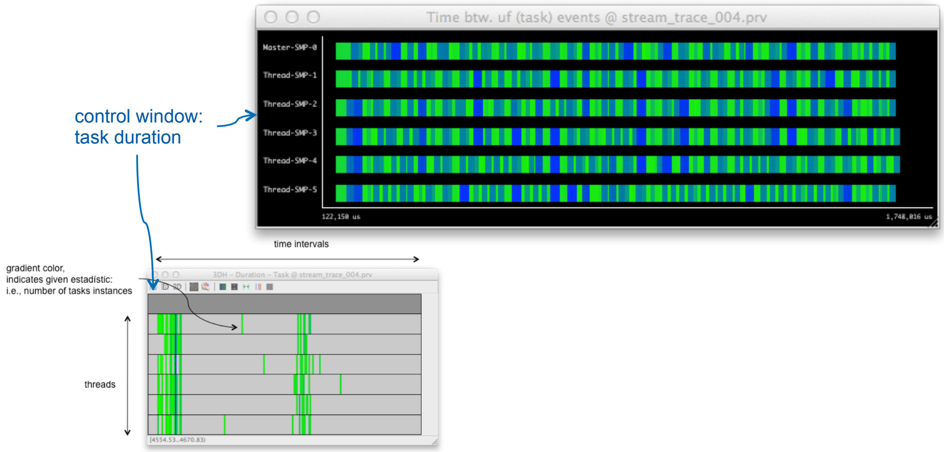
threads



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tasks duration histogram

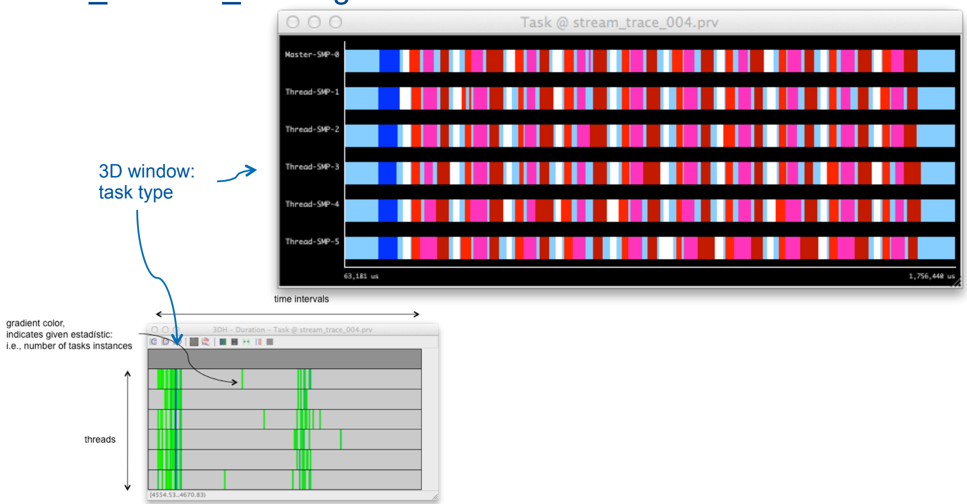
- 3dh_duration_task.cfg



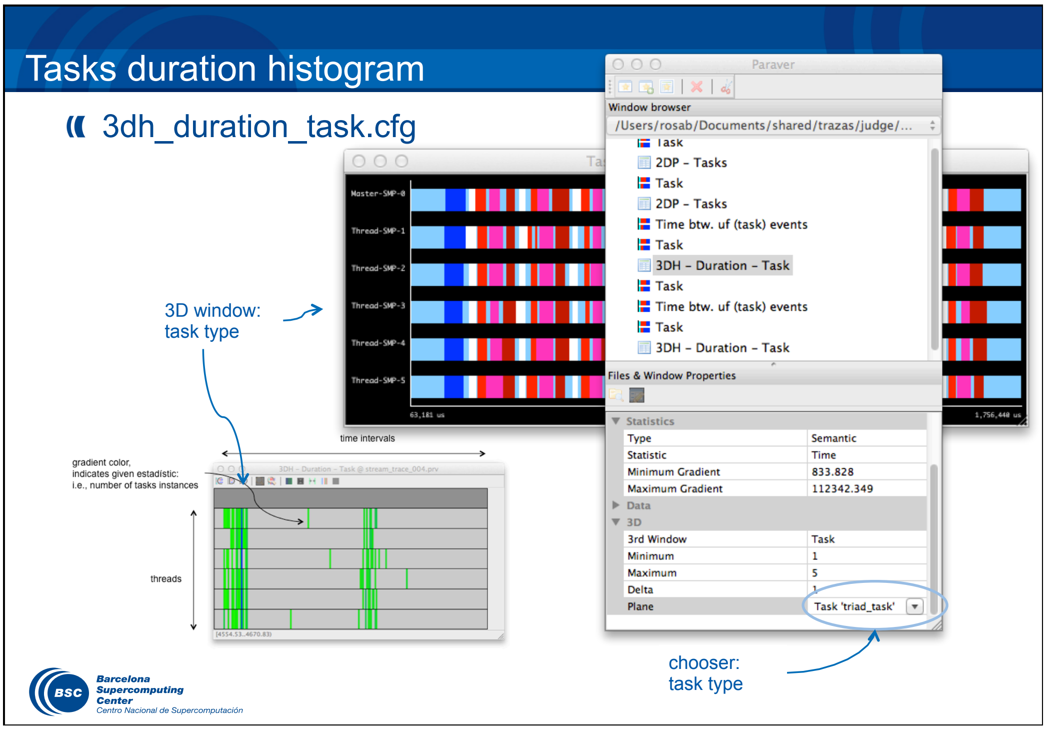
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

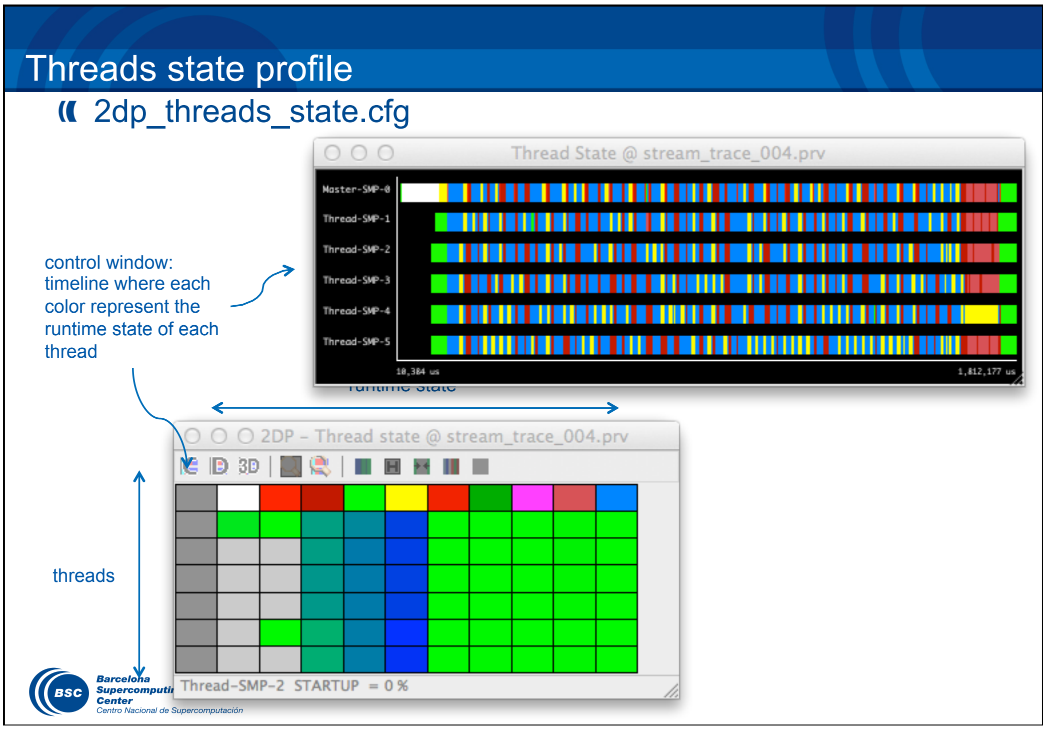
Tasks duration histogram

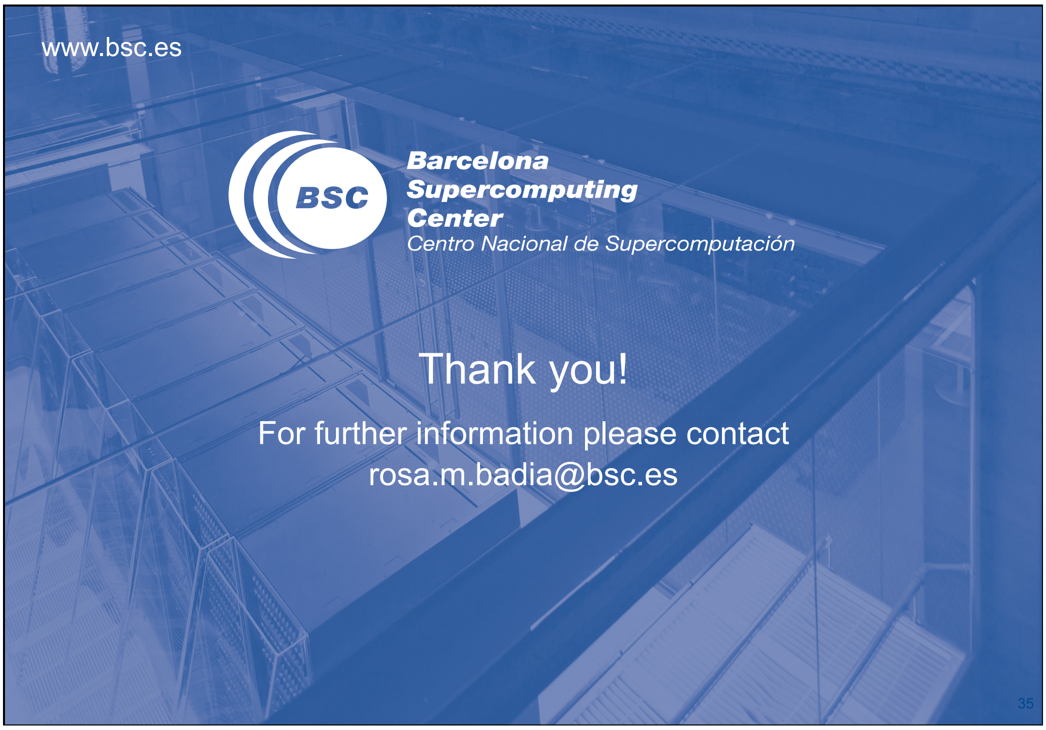
- 3dh_duration_task.cfg



Barcelona Supercomputing Center
Centro Nacional de Supercomputación


Tasks duration histogram
3dh_duration_task.cfg
3D window:
task type
gradient color,
indicates given estadistic:
i.e., number of tasks instances
threads
time intervals
Paraver
Window browser
2DP – Tasks
Task
Time btw. uf (task) events
3DH – Duration – Task
Files & Window Properties
Statistics
Type
Semantic
Statistic
Time
Minimum Gradient
833.828
Maximum Gradient
112342.349
Data
3D
3rd Window
Task
Minimum
1
Maximum
5
Delta
Plane
Task 'triad_task'
chooser:
task type
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación


Threads state profile
2dp_threads_state.cfg
Thread State @ stream_trace_004.prv
Master-SMP-0
Thread-SMP-1
Thread-SMP-2
Thread-SMP-3
Thread-SMP-4
Thread-SMP-5
control window:
timeline where each
color represent the
runtime state of each
thread
2DP – Thread state @ stream_trace_004.prv
threads
Thread-SMP-2 STARTUP = 0 %
Barcelona
Center
Centro Nacional de Supercomputación


www.bsc.es
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
Thank you!
For further information please contact
rosa.m.badia@bsc.es
35