Improving the Dynamic Creation of Processes in MPI-2

Márcia C. Cera, Guilherme P. Pezzi, Elton N. Mathias,
Nicolas Maillard, Philippe O. A. Navaux

{mccera, pezzi, enmathias, nicolas, navaux}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul - Brazil

This work has been partially supported by HP Brazil

IV Workshop PPD/UFRGS
Porto Alegre, Brasil - Julho 2006

Outline

Outline

Motivation

- MPI-2 Norm (1997)
  - Dynamic creation of processes
  - Remote Memory Access - RMA
  - Parallel I/O
- The **norm does not define** any way to schedule dynamic processes
  - Which processor will receive a new process?
  - Inside a processor, in which order will the processes run?

Goal

- Offer on-line scheduling to MPI-2 programs
- Aiming at **load balance**
  - Number of processes created dynamically on each processor.

Motivation

- MPI-2 Norm (1997)
  - **Dynamic creation of processes**
  - Remote Memory Access - RMA
  - Parallel I/O
- The **norm does not define** any way to schedule dynamic processes
  - Which processor will receive a new process?
  - Inside a processor, in which order will the processes run?

Goal

- Offer on-line scheduling to MPI-2 programs
- Aiming at **load balance**
  - Number of processes created dynamically on each processor.

Motivation

- MPI-2 Norm (1997)
  - **Dynamic creation of processes**
  - Remote Memory Access - RMA
  - Parallel I/O
- The **norm does not define** any way to schedule dynamic processes
  - Which processor will receive a new process?
  - Inside a processor, in which order will the processes run?

Goal

- Offer on-line scheduling to MPI-2 programs
- Aiming at **load balance**
  - Number of processes created dynamically on each processor.

Motivation

- MPI-2 Norm (1997)
  - **Dynamic creation of processes**
  - Remote Memory Access - RMA
  - Parallel I/O
- The **norm does not define** any way to schedule dynamic processes
  - Which processor will receive a new process?
  - Inside a processor, in which order will the processes run?

Goal

- Offer on-line scheduling to MPI-2 programs
- Aiming at **load balance**
  - Number of processes created dynamically on each processor.

MPI Distributions and MPI-2

- Some MPI distributions have started providing MPI-2 for a few years.
  - LAM-MPI, since the 2000's;
    - Also implements some tools to manage the dynamic entry of resources and their exit (`lamgrow`/`lamshrink`);
    - Based on daemons.
  - MPI-CH, since Jan., 2005.
  - HP-MPI, since Dec. 2005.
- MPI could almost be used on Grids:
  - It needs support for heterogeneity (MPI-G2);
  - It needs support for Fault-Tolerance (MPI-CHv2, MPI-FT).
- Open-MPI is a merge of LAM and MPI-FT, which could bring everything in a single distribution!

Outline

Dynamic Creation of Processes

- `MPI_Comm_spawn(char* command, char** argv, int maxprocs, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm, int *errcodes)`

Dynamic Creation of Processes

- `MPI_Comm_spawn(char* command, char** argv, int maxprocs, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm, int *errcodes)`
  - Name of a **MPI executable**
    - A multi-process program with `MPI_Init` and `MPI_Finalize`
    - This executable will be the **child** program
  - **Arguments** of executable program
    - Command line parameters of the child program
  - **Number of processes** that will execute the child program

Dynamic Creation of Processes

- `MPI_Comm_spawn(char* command, char** argv, int maxprocs, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm, int *errcodes)`
  - **Startup hints**
  - For resource allocation LAM-MPI offers `MPI_Info` **keys** set by `MPI_Info_set`
    - `lam_spawn_file` - defines a file appschema with available nodes
    - `lam_spawn_sched_round_robin` - uses the LAM nodes making a Round-Robin distribution that starts on a determined node
    - `MPI_INFO_NULL` - makes a Round-Robin distribution start on node with the lowest rank

Dynamic Creation of Processes

- `MPI_Comm_spawn(char* command, char** argv, int maxprocs, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm, int *errcodes)`
  - **Parent intracommunicator**
    - Children discover this intracommunicator by `MPI_Comm_get_parent`
  - **Child intercommunicator** containing spawned processes
    - By this intracommunicator the parent exchanges messages with its children

The Fibonacci Example with MPI-2

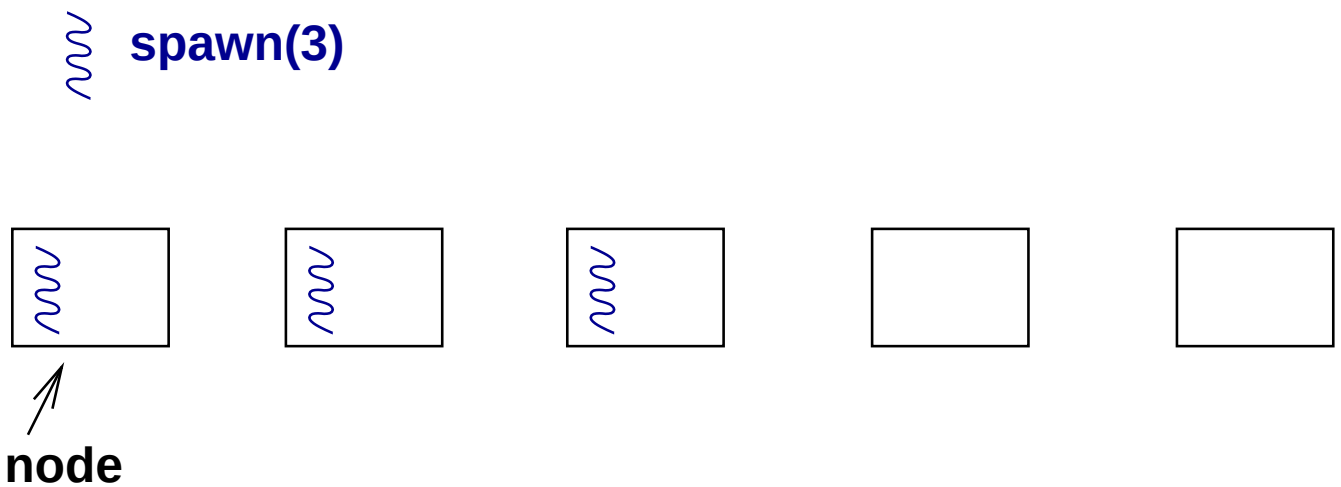
MPI-2 Code of the "fibo.c" program:

```
MPI_Comm_get_parent(&parent);
if (n < 2) {
   MPI_Isend (&n, 1, MPI_LONG, 0, 1, parent, &req);
}
else{
   sprintf (argv[0], "%ld", (n - 1));
   MPI_Comm_spawn ("Fibo", argv, 1, local_info, myrank, MPI_COMM_SELF, &children_comm[0],
                                                                       errcodes);
   sprintf (argv[0], "%ld", (n - 2));
   MPI_Comm_spawn ("Fibo", argv, 1, local_info, myrank, MPI_COMM_SELF, &children_comm[1],
                                                                       errcodes);
   MPI_Recv (&x, 1, MPI_LONG, MPI_ANY_SOURCE, 1, children_comm[0], MPI_STATUS_IGNORE);
   MPI_Recv (&y, 1, MPI_LONG, MPI_ANY_SOURCE, 1, children_comm[1], MPI_STATUS_IGNORE);
   fibn = x + y;
   MPI_Isend (&fibn, 1, MPI_LONG, 0, 1, parent, &req);
}
MPI_Finalize ();
```

Outline

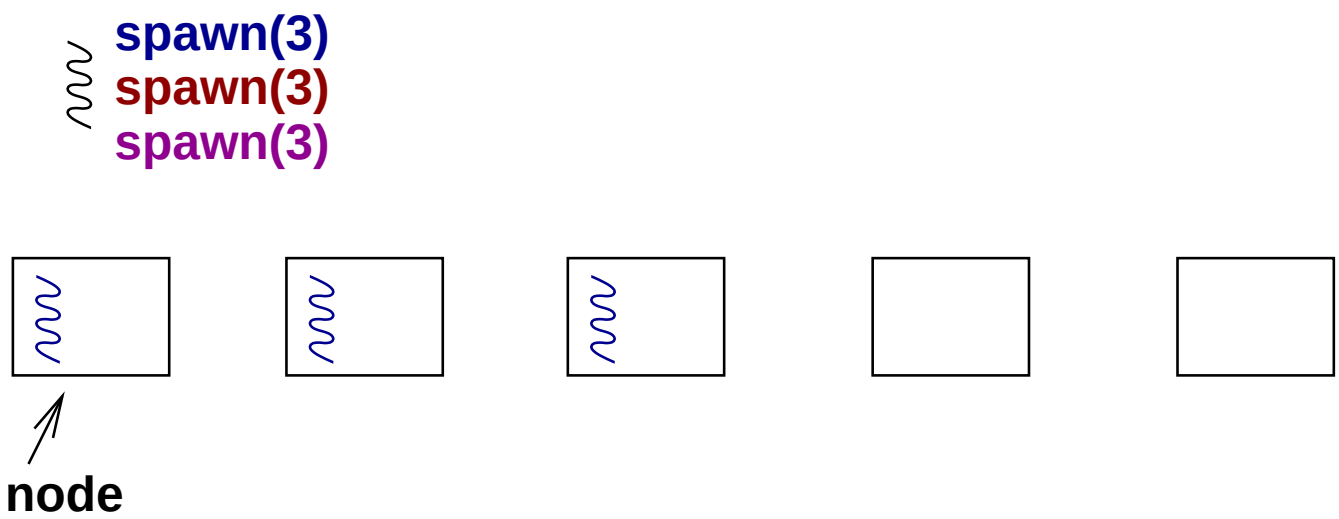
How to Schedule the Spawned Processes

- LAM-MPI provides a **Round-Robin mechanism**
  - It distributes `maxprocs` processes, **one by node available**
  - `MPI_Info_set(info, "lam_spawn_sched_round_robin", node)`



How to Schedule the Spawned Processes

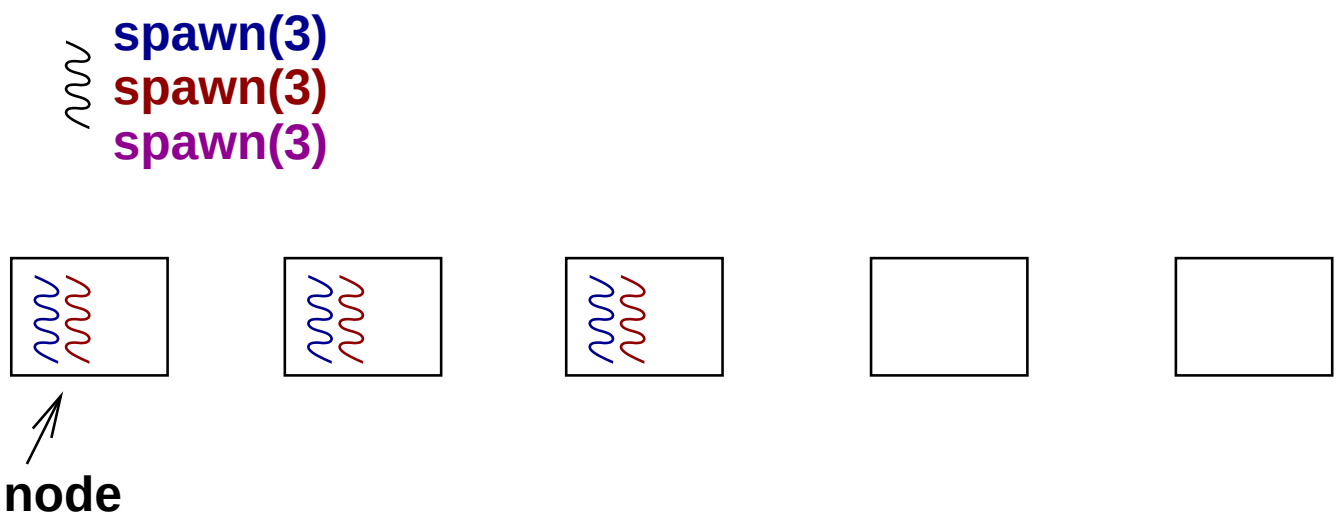
- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

How to Schedule the Spawned Processes

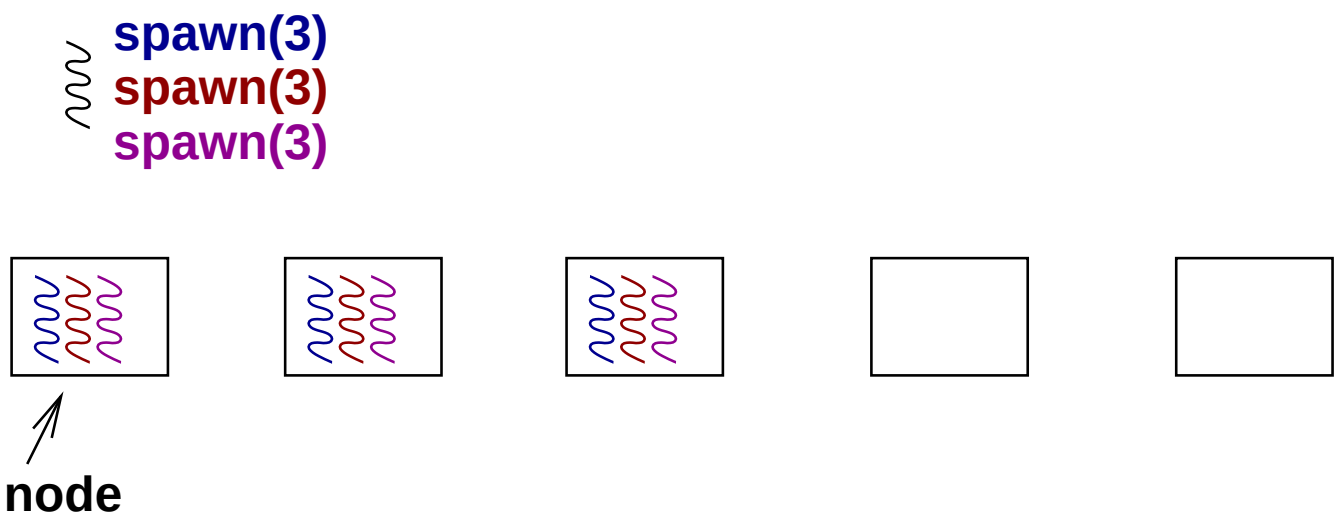
- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

How to Schedule the Spawned Processes

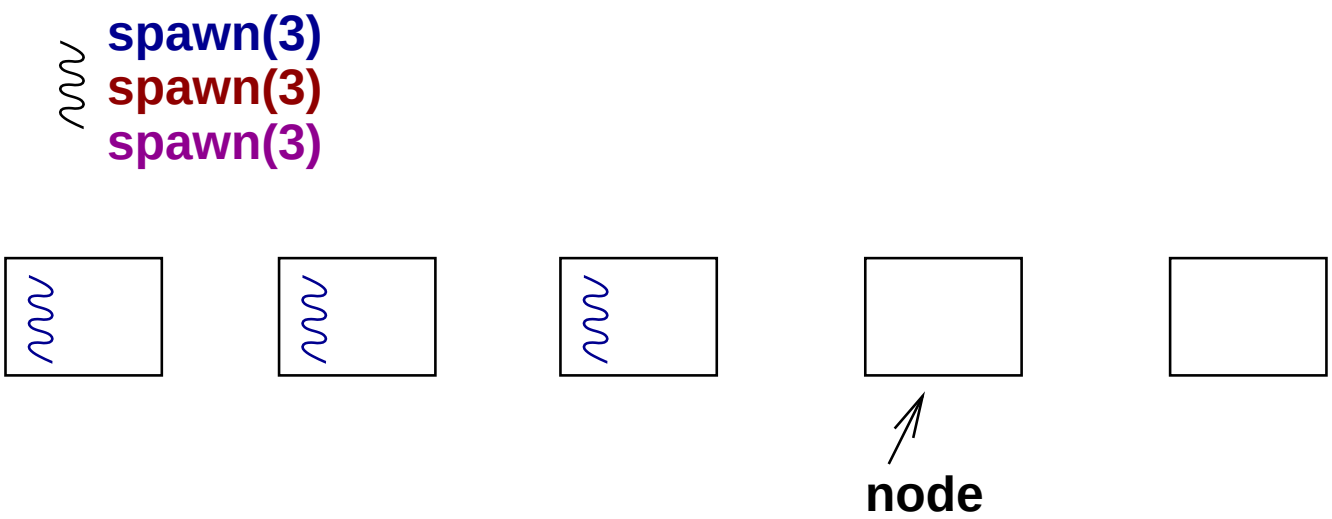
- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

How to Schedule the Spawned Processes

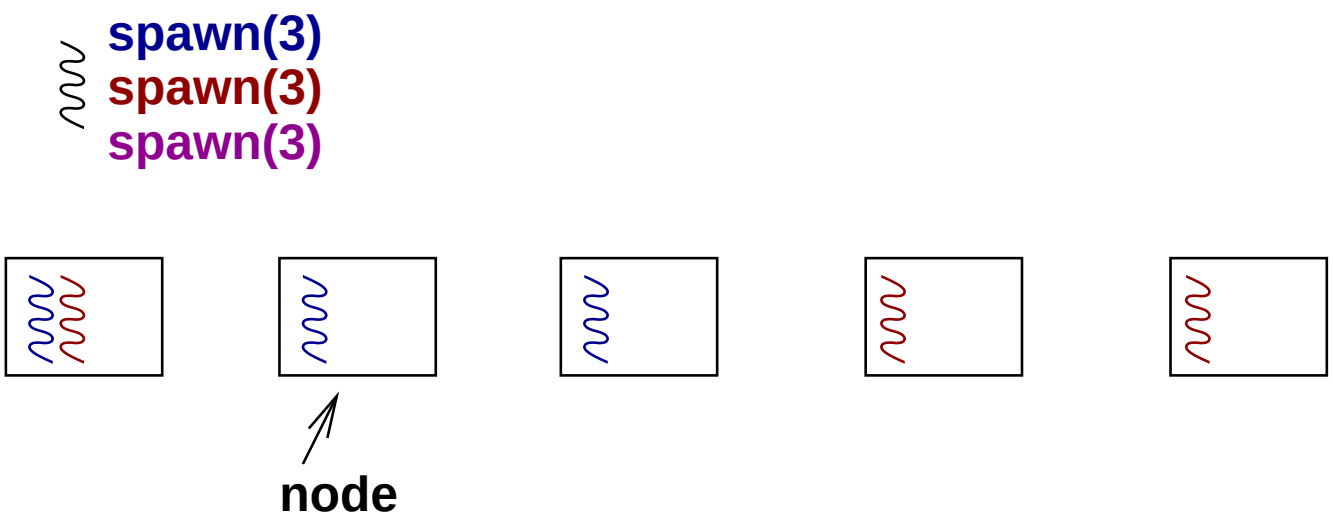
- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

How to Schedule the Spawned Processes

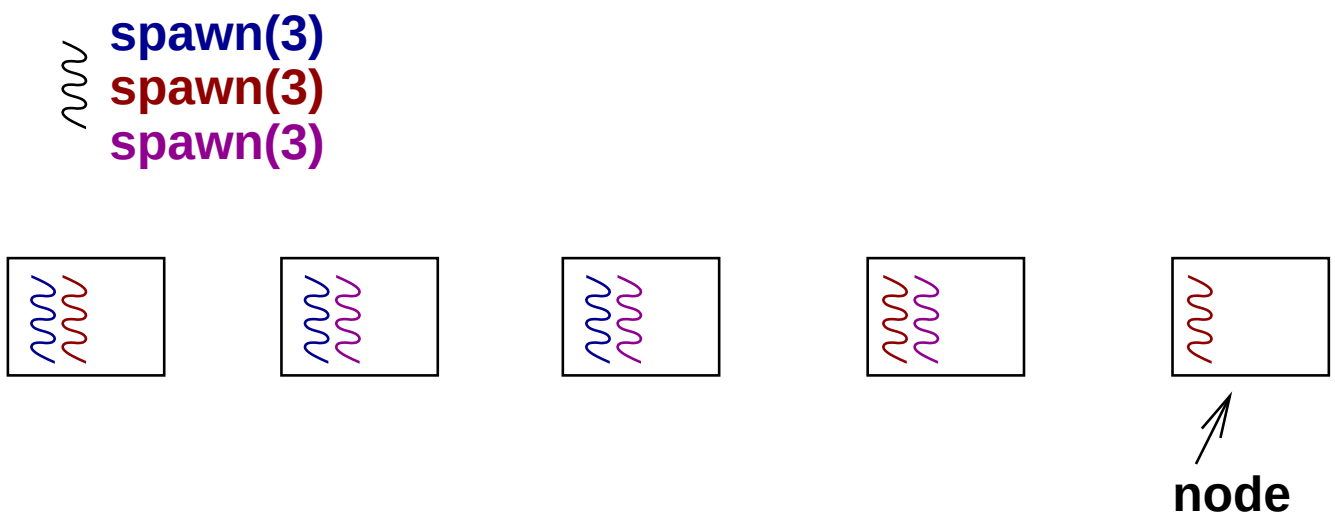
- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

How to Schedule the Spawned Processes

- If there is a **loop** structure spawning processes



  - The distribution is **not balanced** - needs to know who has received processes
  - In a **distributed case** the problem is even bigger

Experimental Tests

- Spawning 20 processes on 5 nodes using single and multiple spawn calls with **LAM Round-Robin mechanism**

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
20 spawns of 1 process	**20**	**0**	**0**	**0**	**0**
1 spawn of 20 processes	4	4	4	4	4

Experimental Tests

- Spawning 20 processes on 5 nodes using single and multiple spawn calls with **LAM Round-Robin mechanism**

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
20 spawns of 1 process	20	0	0	0	0
1 spawn of 20 processes	**4**	**4**	**4**	**4**	**4**

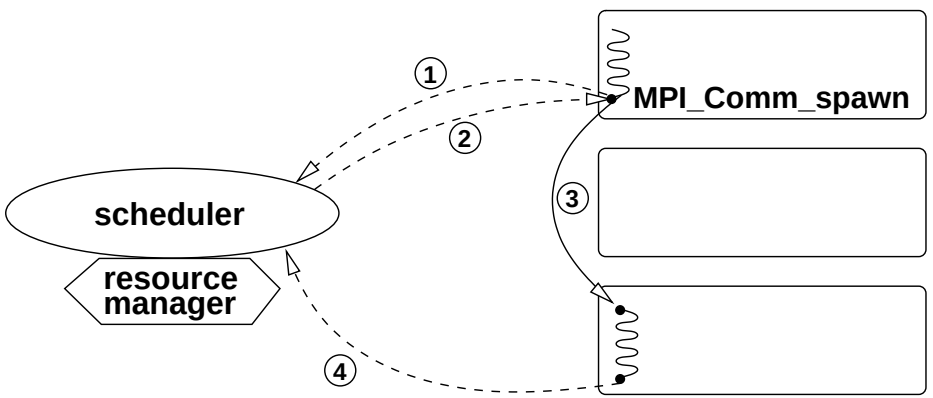
Outline

The Proposed Scheduler

- A centralized deamon receives messages from re-defined MPI-2 primitives and takes the **scheduling decisions**
- The usage of pre-compilation redefinition is simple and portable.
  - the end-user just has to re-compile his program.

Interactions between MPI-2 processes and scheduler

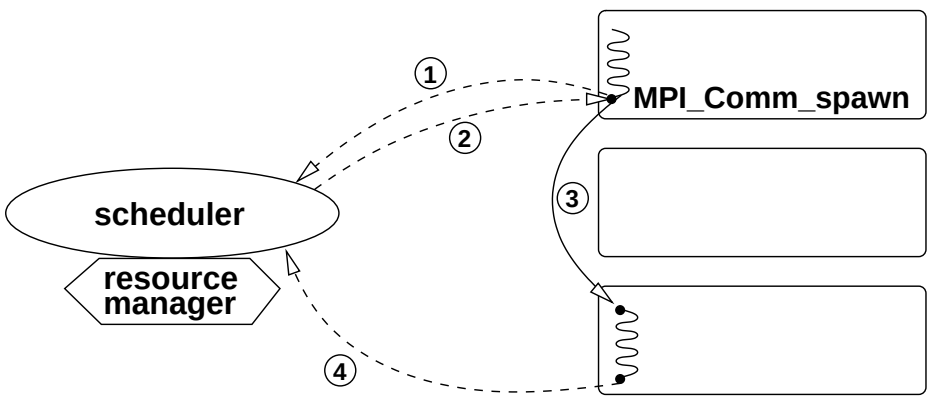
- **Interactions between MPI-2 processes and scheduler**



1. Notification of processes creation
2. Physical location is returned to parent
3. The child is physically spawned
4. Notification of processes completion

Interactions between MPI-2 processes and scheduler

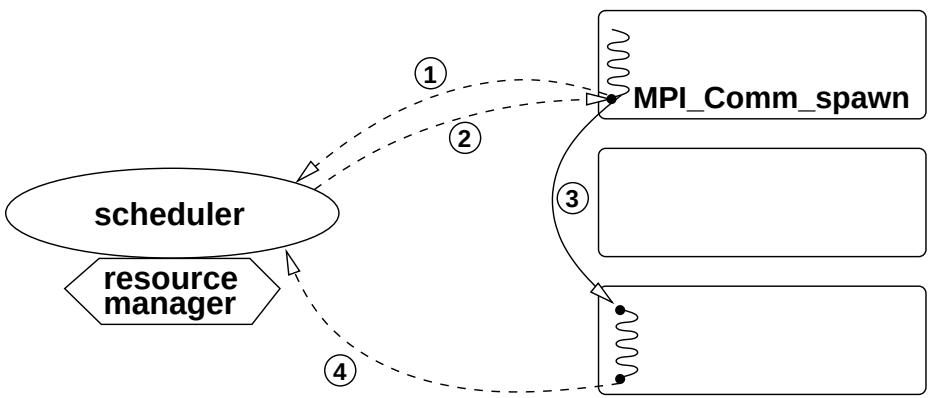
- **Interactions between MPI-2 processes and scheduler**



1. Notification of processes creation
2. Physical location is returned to parent
3. The child is physically spawned
4. Notification of processes completion

Interactions between MPI-2 processes and scheduler

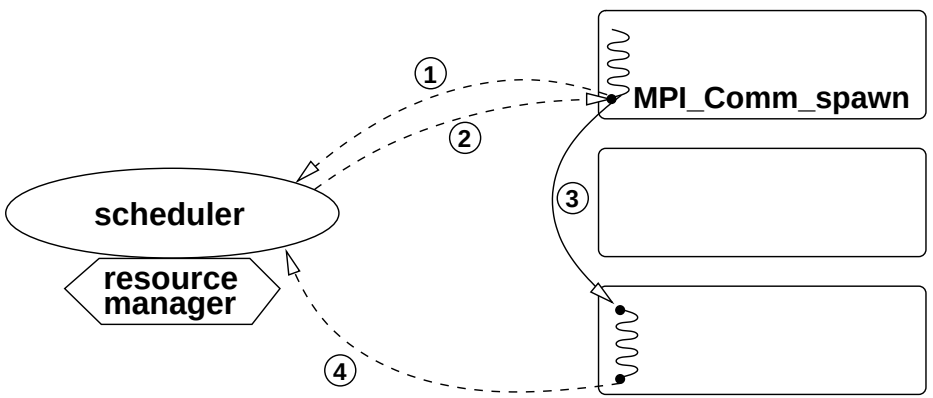
- **Interactions between MPI-2 processes and scheduler**



1. Notification of processes creation
2. Physical location is returned to parent
3. The child is physically spawned
4. Notification of processes completion

Interactions between MPI-2 processes and scheduler

- **Interactions between MPI-2 processes and scheduler**



1. Notification of processes creation
2. Physical location is returned to parent
3. The child is physically spawned
4. Notification of processes completion

Scheduling Heuristics

- Scheduling heuristics can be applied in **two levels**:
  - Scheduling **processes into resources**
  - **Priorizing the execution of ready processes** into a resource (actually it is left under OS responsibility)
- The scheduler implements our **Round-Robin mechanism**
  - The scheduler knows the last resource used
  - $new\_resource = (last\_resource + 1)\%total\_resources$
- Scheduling decisions according to the **resources load information**
  - **Load Monitor** – collect the load information
  - **Resource Manager** – coordinate of the load monitors and keeping a list of resources updated

The Fibonacci Test-case

- Comparing different schedules: **number of processes** spawned on each node

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
fib(6) with LAM standard scheduler	25	0	0	0	0
fib(6) with embedded scheduler	8	4	8	2	3
fib(6) with proposed scheduler	5	5	5	5	5

The Fibonacci Test-case

- Comparing different schedules: **number of processes** spawned on each node

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
fib(6) with LAM standard scheduler	**25**	**0**	**0**	**0**	**0**
fib(6) with embedded scheduler	8	4	8	2	3
fib(6) with proposed scheduler	5	5	5	5	5

- Sets always the same first Round-Robin node

The Fibonacci Test-case

- Comparing different schedules: **number of processes** spawned on each node

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
fib(6) with LAM standard scheduler	25	0	0	0	0
fib(6) with embedded scheduler	**8**	**4**	**8**	**2**	**3**
fib(6) with proposed scheduler	5	5	5	5	5

- Sets always the same first Round-Robin node
- Sets the first Round-Robin node to neighbor

The Fibonacci Test-case

- Comparing different schedules: **number of processes** spawned on each node

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
fib(6) with LAM standard scheduler	25	0	0	0	0
fib(6) with embedded scheduler	8	4	8	2	3
fib(6) with proposed scheduler	**5**	**5**	**5**	**5**	**5**

- Sets always the same first Round-Robin node
- Sets the first Round-Robin node to neighbor
- Sets the first Round-Robin node as global information

The Fibonacci Test-case

- LAM-MPI imposes a **file description limitation**
- Restrictions about the **maximum number of processes running** in a resource
- The Round-Robin mechanism proposed makes it possible to compute the 13th Fibonacci number

The Fibonacci Test-case

- LAM-MPI imposes a **file description limitation**
- Restrictions about the **maximum number of processes running** in a resource
- The Round-Robin mechanism proposed makes it possible to compute the 13th Fibonacci number

	Node 1	Node 2	Node 3	Node 4	Node 5	Total Number of Processes
fib(13)	151	151	151	150	150	**753**

2nd Test-Case: an Irregular Computation

- Computation of **prime numbers in a recursive search** – like Fibonacci program but **irregular**
- This program is CPU-intentive and a good load balance should impact the running time.
- Intervals range between 1 and 20 millions

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
LAM standard scheduler	39	0	0	0	0
proposed scheduler	8	8	8	8	7

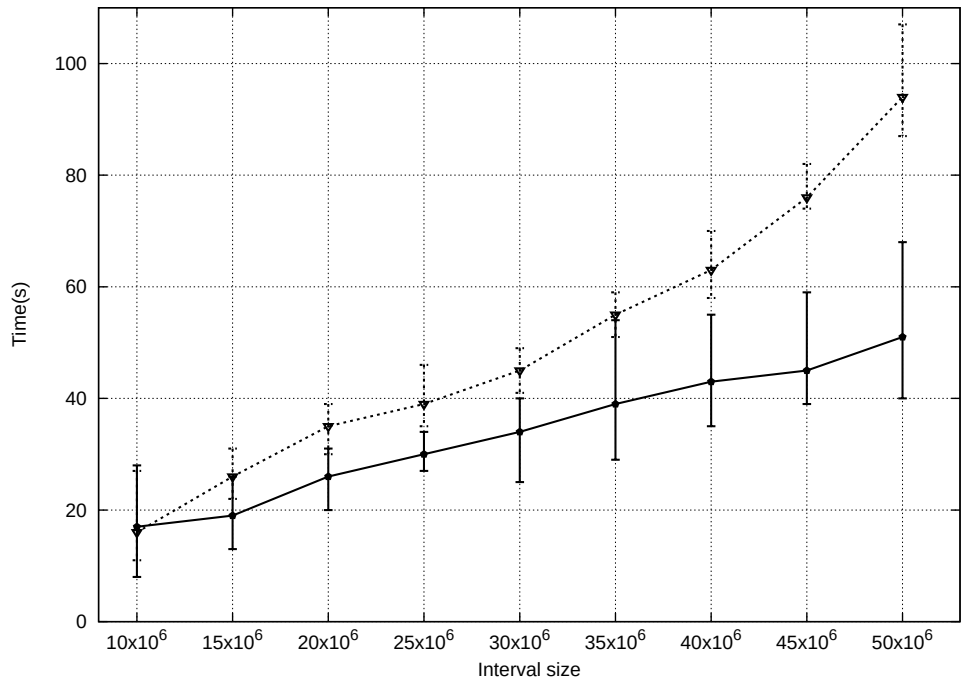
2nd Test-Case: an Irregular Computation

- Computation of **prime numbers in a recursive search** – like Fibonacci program but **irregular**
- This program is CPU-intentive and a good load balance should impact the running time.
- Intervals range between 1 and 20 millions

Environment	Node 1	Node 2	Node 3	Node 4	Node 5
LAM standard scheduler	39	0	0	0	0
proposed scheduler	**8**	**8**	**8**	**8**	**7**

- **Good load balance** with proposed scheduler
  - 181.15*s* with LAM standard scheduler
  - 46.12*s* with proposed scheduler

List-scheduleing – Using Dynamic Load Information



- Round-Robin strategy (dotted line) and List scheduling (solid line)

Outline

Conclusion

- This work aims to simplify the **on-line scheduling** of MPI-2 programs
  - Interest for dynamic platforms!
- The native LAM implementation is not efficient, due to a simple scheduling strategy.
  - A simple prototype led to clear improvements

Next Steps

- To implement the proposed scheduler **inside of LAM-MPI distribution**
- Tests with real-world applications
  - Branch & Bound, linear systems solving, etc...
- To **distribute** the centralized scheduler
- To implement a **workstealing strategy** for the distributed scheduler

Improving the Dynamic Creation of Processes in MPI-2

Márcia C. Cera, Guilherme P. Pezzi, Elton N. Mathias,
Nicolas Maillard, Philippe O. A. Navaux

{mccera, pezzi, enmathias, nicolas, navaux}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul - Brazil

This work has been partially supported by HP Brazil

IV Workshop PPD/UFRGS
Porto Alegre, Brasil - Julho 2006