

A Flexible Thread Scheduler for Hierarchical Multiprocessor Machines

Samuel Thibault



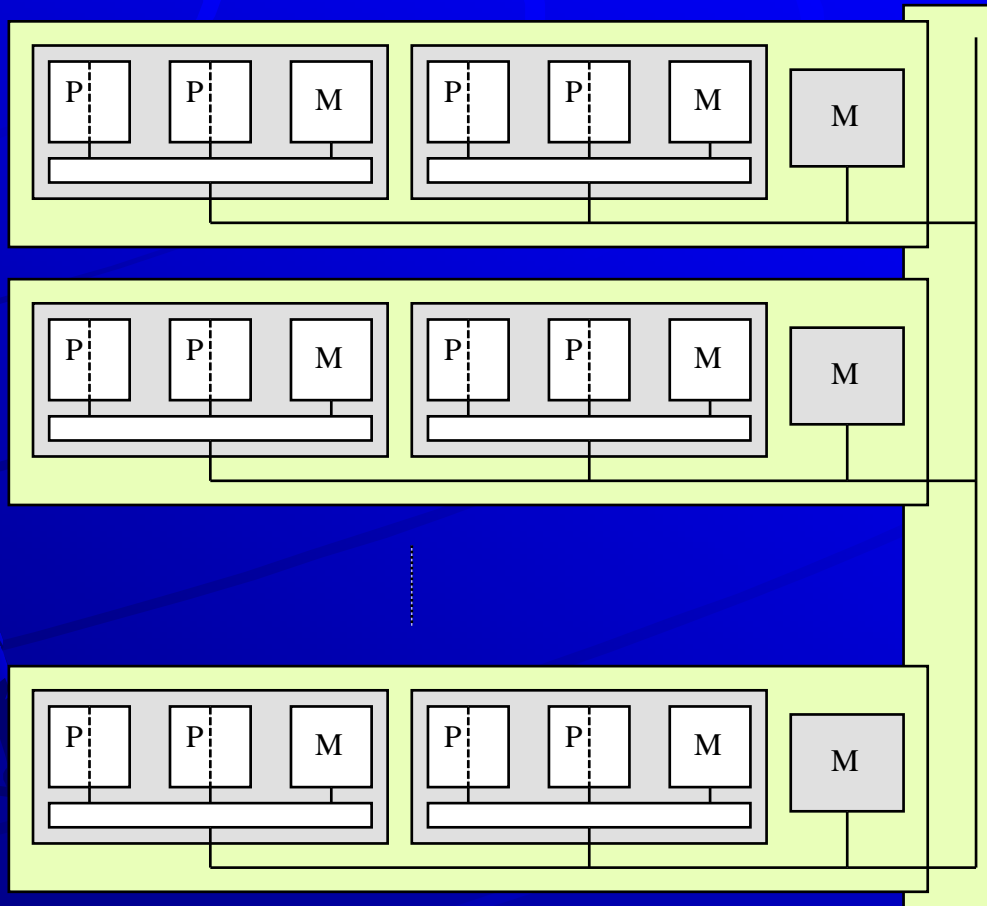
INRIA Runtime Project
LaBRI, Université de Bordeaux 1
FRANCE

Current trend is increasing hierarchy within multiprocessor machines

- ◆ NUMA
- ◆ Multi-Core
- ◆ HyperThreading
- ◆ ...

◆ Intel, IBM, Sun, ...

Future machines



✦ Not only NUMA factor

The challenge is how to schedule applications efficiently

◆ Performance

- Affinities between threads and memory taken into account

◆ Flexibility

- Execution easily controlled by applications

◆ Portability

- Applications adapted to any new machine

Thread scheduling on hierarchical machines

Current approaches:

- ◆ Predetermined
- ◆ Opportunist
- ◆ Negotiated



Predetermined scheduling

◆ Preliminary computation of

- Data placement
- Threads scheduling

◆ Execution

- Both threads and data are fixed in the operating system, explicit scheduling is used

◆ Example: PaStiX, a linear algebra solver

✓ Excellent performance

✗ Not always possible: strongly irregular problems...

Opportunist scheduling

- ◆ Several greedy algorithms (Self Scheduling)
 - Single task list
 - One task list per processor
 - Groups of task lists
- ◆ Used in nowadays' operating systems:
 - Linux, Solaris, FreeBSD, Windows, ...
- ✓ Portability: scale well
- ✗ Performance: don't have direct affinity information

Negotiated scheduling

✦ Language extensions

- OpenMP, HPF, UPC, ...
- The compiler inserts code to manage threads

✓ Performance: adapts itself to the machine

✗ Flexibility: not generic enough

✦ Operating System support

- liblgroup (Solaris), libnuma (Linux)
- Programmers are left to themselves

✓ Freedom for programmers

✗ Flexibility: needs rewriting a scheduler

Issues

- ◆ Which scheduling strategies?
- ◆ Which affinities between threads and data?
- ◆ In general, programmers know them
 - How to let programmers express that?
- ➔ We propose a flexible scheduler allowing to experiment several strategies

Bubble Scheduling

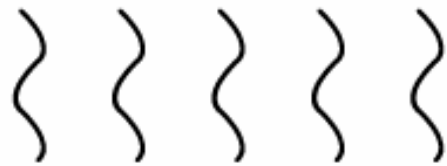
An approach which is both
predetermined, opportunist and
negotiated



Application structure

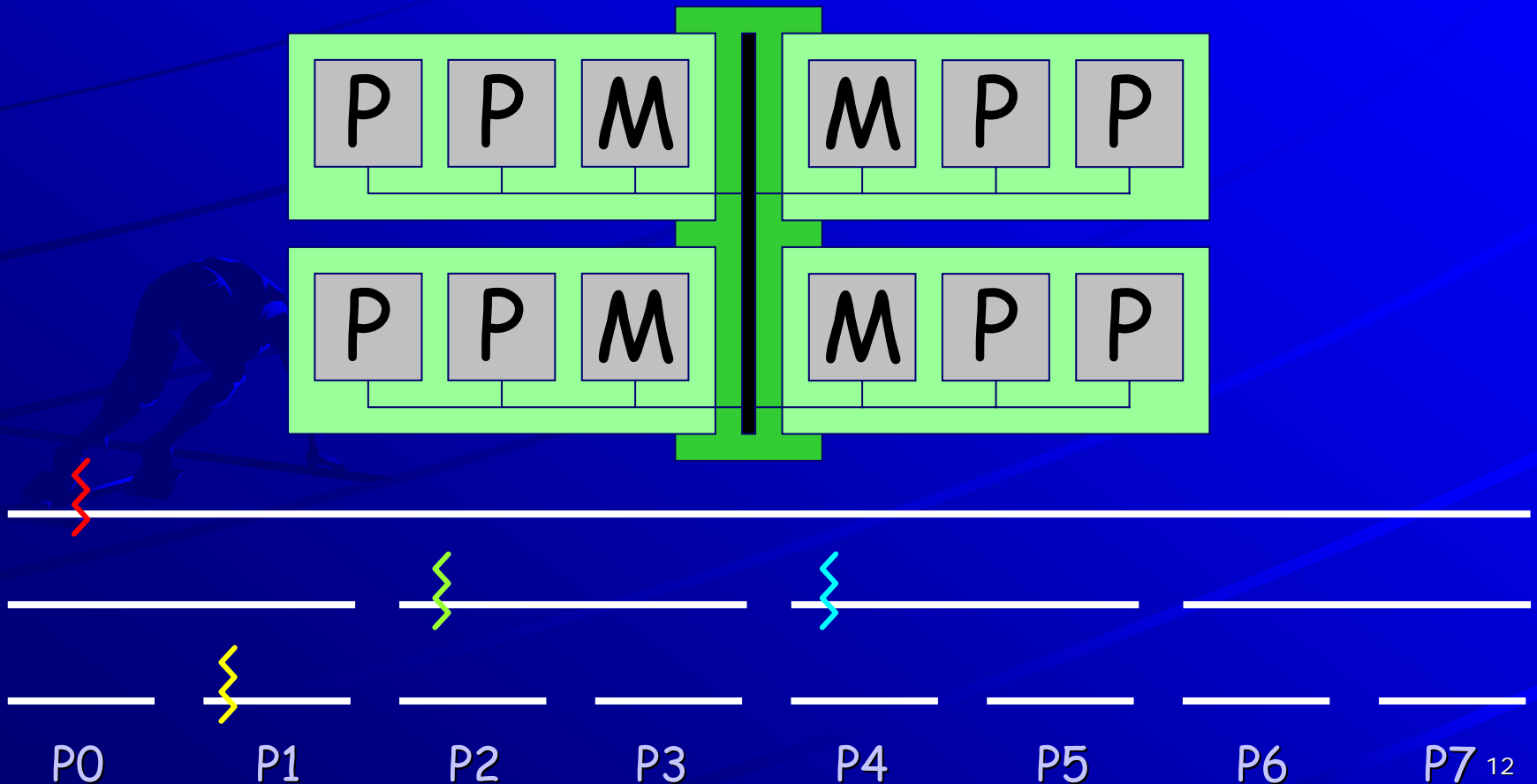
✦ Thread affinities are expressed through bubbles

- Data sharing
- Collective operations
- ...

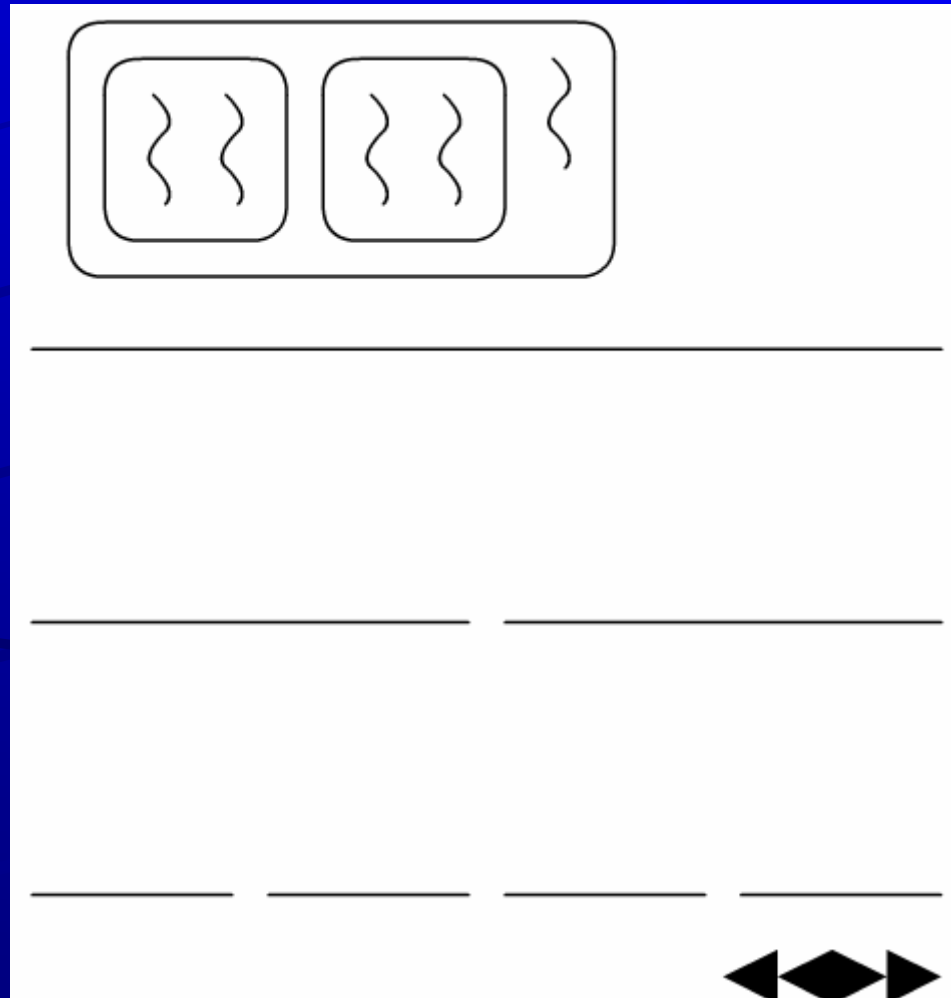


Scheduling domains

- ◆ A hierarchy of task lists

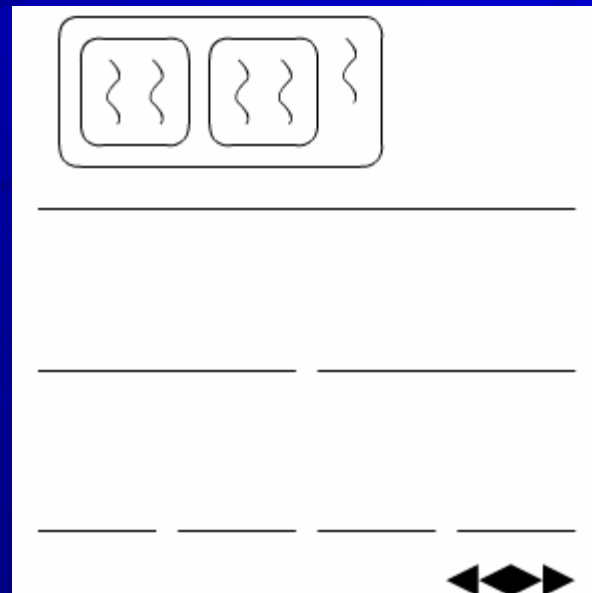


A bubble scheduler



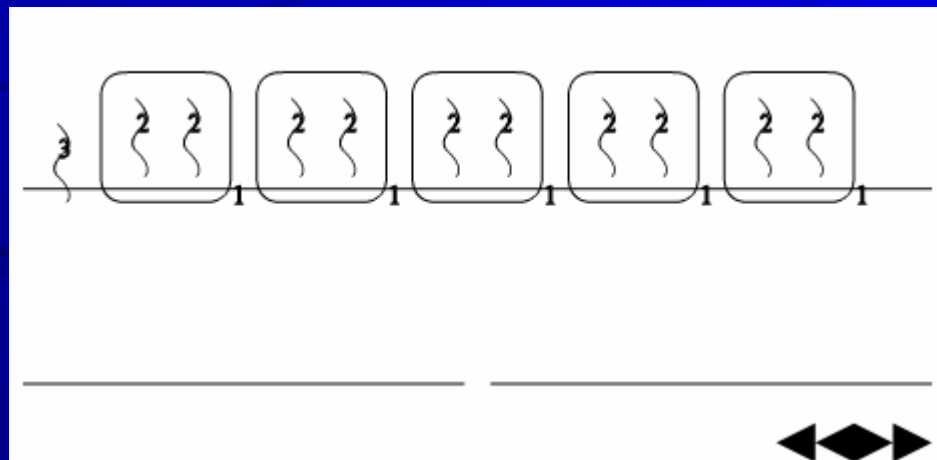
A bubble scheduler

- ◆ Threads and bubbles evolve on task lists
 - Processors "pull" them in their direction
 - Expressed affinities are taken into account
 - Burst levels set by the programmer



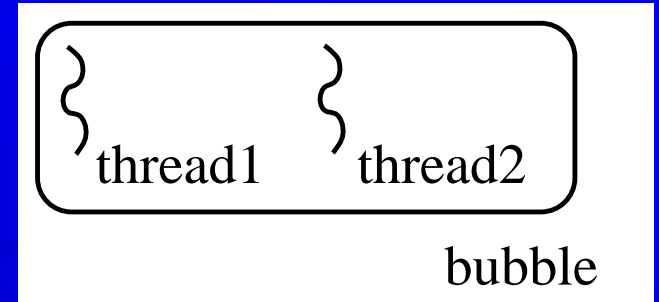
Priorities and bubble regeneration

- ◆ Priorities: controlling scheduling
- ◆ Periodical regeneration: load regulation



API

```
marcel_t thread1, thread2;  
marcel_bubble_t bubble;  
  
marcel_bubble_init(&bubble);
```



```
marcel_create_dontsched(&thread1, NULL, f1, param1);  
marcel_create_dontsched(&thread2, NULL, f2, param2);  
  
marcel_bubble_inserttask(&bubble, &thread1);  
marcel_wake_up_bubble(&bubble);  
marcel_bubble_inserttask(&bubble, &thread2);
```

Implementation

- ◆ All these mechanisms are implemented within a two-level thread scheduler (called Marcel, part of the PM2 software)
 - Portable
 - Efficient
 - Extensible

Implementation issues

- ◆ Decentralized scheduling
 - Making decisions
- ◆ Task list hierarchy
 - Taking priorities into account
 - Locking
- ◆ Bubbles regeneration
 - Preempting running threads

Micro-benchmark

- ✦ On HyperThreaded bi - PIV Xeon
 - 3 hierarchical levels

	"find_next" (cycles)	Locking + "switch_to" (cycles)
Marcel (original)	495	233
Marcel bubbles	665	395
NPTL (Linux 2.6)	1790	3930

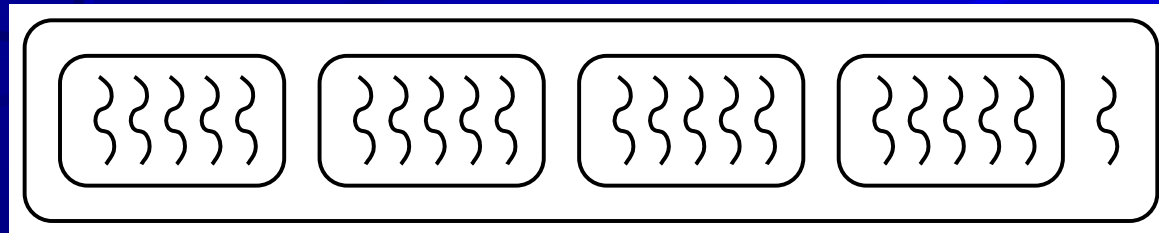
Application: conduction and advection

- ◆ Heat conduction and advection simulation
- ◆ Mesh divided in stripes, according to the number of processors
- ◆ Target machine: Bull NovaScale
 - NUMA machine composed of 4 blocks, each featuring:
 - ◆ 4 Itanium II processors
 - ◆ 16 GB of memory
 - NUMA factor: approx. 3

Application: conduction and advection

◆ Three approaches

- Simple: opportunist scheduling
- Fixed: threads fixed on processors
- Bubbles: recursive bubbles strategy



Results (16 processors)

	Conduction		Advection	
	Time (s)	Speed up	Time (s)	Speed up
Sequential	250.2		16.13	
Simple	23.65	10.58	1.77	9.11
Fixed	15.82	15.82	1.30	12.40
Bubbles	15.84	15.80	1.30	12.40

Conclusion

- ✦ Current trend is toward hierarchical machines
 - ➔ Scheduling issues
- ✦ Bubbles let applications express themselves
 - No supposition about the machine
- ✓ Portable, flexible and efficient
- ✓ First results are promising
- ✓ Try different strategies

On going work, perspectives

- ◆ Tools for observing and analyzing scheduling
 - Study different strategies
- ◆ More affinity indications
 - Tune bubble elasticity
- ◆ Memory allocations
- ◆ An even more generic bubble scheduler
 - Automatic higher or lower burst levels
 - Automatic rebalancing

Offline replay

o



Questions?

[http://runtime.futurs.inria.fr/Runtime/Download/
pm2-stable-mars-2005.tar.bz2](http://runtime.futurs.inria.fr/Runtime/Download/pm2-stable-mars-2005.tar.bz2)

