

Dynamic Load-Balancing Algorithm Porting on MIMD Machines

- Francisco Junqueira Muniz
- Nuclear Technology Development Center
CDTN
- Belo Horizonte - Brazil - muniz@cdtn.br

This presentation

- Describes the porting strategies and the implementation of a dynamic load-balancing mechanism over the PVM library;
- some results that validate the usefulness of the load-balancing system are presented.

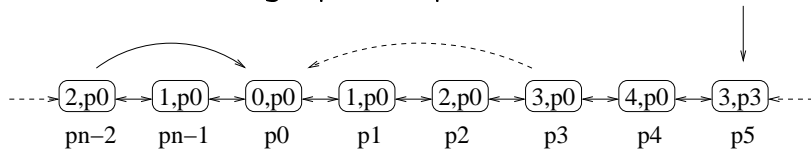
- A comprehensive survey on existing resources management approaches is found in a literature review.
- D. G. Feitelson, Job Scheduling in Multiprogrammed Parallel Systems, IBM T. J. Watson Research Center, 1997, August, IBM RC 19790, 175 pages (638 references).

Gradient Model (GM) mechanism

- Is a locally-distributed scheme with a global placement;
- periodic exchange of load-balancing information is restricted to a small and pre-specified subset of other nodes in the cluster (therefore a potentially scalable mechanism);
- the processor status is determined by the local processor availability, i.e. a measure of the loading level of the processor, and the status of its neighbour nodes - this status represents the logical distance (number of intermediate neighbour nodes plus one) to the nearest idle node in the cluster;

- the node status information is propagated, from lightly-loaded to over-loaded nodes, through a message interchanging among neighbour nodes;
 - the system resource availability can be requested at arbitrary execution times in any node.
-
- F. C. H. Lin and M. R. Keller. Gradient model: a demand-driven load balancing scheme. In *IEEE Conf. on Distributed Systems*, pages 329–336, 1986.

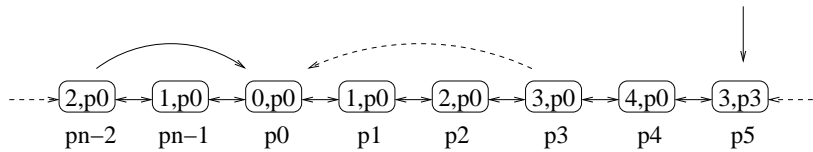
GM graphic representation



Values (x,y) inside a rectangle with round corners represent: (x) the logical distance (number of intermediate neighbour processors plus one) to nearest idle node and (y) the identification of the idle processor.

GM scheme drawbacks

- Out-of-date information;
- overflow effect.
- S. Nishimura and T. L. Kunii. A decentralized dynamic scheduling scheme for transputers networks. In T. L. Kunii and D. May, editors, *Proc. of the 3rd Transputer/occam Int. Conf.*, pages 181–194, Tokyo, Japan, May 1990.



Out-of-date status with potential for overflow. Processor p_6 has out-of-date information which records processor p_3 as idle. Processor p_{n-2} and processor p_3 simultaneously decide to make a placement on processor p_0 .

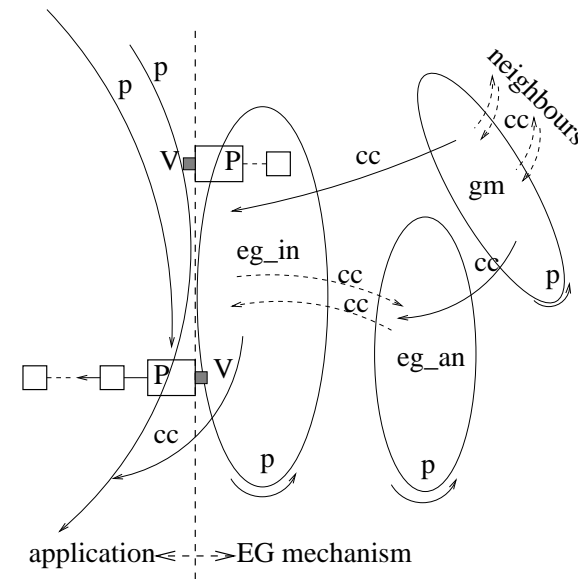
Extended Gradient algorithm

- The EG method interrogates the remote processor identified by the GM scheme as lightly-loaded to confirm that it is still available and reserves it to receive the new processing demand.
- F. J. Muniz and E. J. Zaluska, Parallel load-balancing: An extension to the gradient model, *Parallel Computing*, 1995, vol 21, n 2, pages 287-301, February.

Hardware and software environment

- Linux/X86 compute nodes;
- 'C' language;
- PVM library;
- valency of two (each PC communicates with two others) was established, therefore organised in logical ring topology to allow exchange of the extended gradient load-balancing information.

Graphic representation of the ported EG mechanism



Objects marked with p represent processes: gm is the gradient mechanism; egm_in and egm_an are extensions of the gradient; others are user processes (applications). Continuous lines, marked with cc, represent communication channels; in particular, objects in dotted line, marked with cc, represent communication facilities between processes in distinct nodes. Objects in rectangular forms represent the semaphore primitives. It is also represented in dashed line, the interface between application and machine facilities (EG mechanism part of the operating system).

Application programmer interface

- `pvm_spawn(proc, ..., 1, "machine_name", ...)`
- `pvm_spawn(proc, ..., 0, "", ...)`

- `pvm_spawn(proc, ..., 1, "egm", ...)`

Performance investigations

- CA (Cyclic Allocation load-balancing policy);
- percentage of improvement derived from the total execution times $(1 - \frac{EG}{CA})$;
- experiments were conducted using 6 nodes and 72 processes;
- Hanoi, Ray-tracing and Queens.

Unbalanced processing demand

process number (pn)	0	1	2	3	4	...
Hanoi ($pn\%3 + 2$)	2	3	4	2	3	...
Queens ($2 \times (pn\%3 + 1)$)	2	4	6	2	4	...
Ray-tracing ($pn\%3 + 2$)	2	3	4	2	3	...

Number of times the same code runs in each process is a function of the process number. In total, **Hanoi** (running over sets of 16 up to 30 disks) and **Ray-tracing** (running over 13 scenes) applications execute the same code 216 times (3×72); **Queens** (15 queens on a 15×15 board) execute 288 times.

	execution time EG/CA (in seconds)	% of improvement
Hanoi	2034/2253	10
Queens	1911/2582	16
Ray-tracing	1798/1787	-1

$$\% \text{ of improvement} = \left(1 - \frac{EG}{CA}\right)$$

The algorithm implemented:

- Is a decentralised (a locally-distributed) mechanism;
- has constant cost ($\log n$) - therefore, 'scalable';
- has global placement space;
- can be easily required by application programmer.

Conclusions

This presentation contains the porting of a dynamic load-balancing mechanism on a distributed-memory MIMD machine. Experiments were conducted and results were presented to make evident that a system-level user-independent dynamic load-balancing mechanism is feasible, practical and can significantly improve performance.

Master_Hanoi Module

```
int main() {
    if (master == PvmParentNotSet)
        ntasks = nprocess;
    else { // receive message from the master
        pvm_recv(master, 0);
        pvm_upkint(&ntasks, 1, 1);
    }
    if (ntasks > 0) {
        pvm_spawn(m_h, ..., "", 1, &tids[0]);
        ntasks--;
        pvm_initsend(PvmDataDefault);
        pvm_pkint(&ntasks, 1, 1);
        pvm_send(tids[0], 0); // send message to ...
        pvm_spawn(s_h, ..., "egm", 1, &tids[1]);
    }
}
```

Slave_Hanoi Module

```
int main() {  
    system("hanoi >> out"); // execute hanoi code  
}
```

Acknowledgements

I am most grateful to the whole CENAPAD-MG/CO ('Centro Nacional de Processamento de Alto Desempenho para Minas Gerais e o Centro-Oeste') group, who made computing facilities available for the benchmark experiments.