Load balancing in dynamic networks by bounded delays asynchronous diffusion

Jacques M. Bahi [1], Sylvain Contassot-Vivier [2], Arnaud Giersch [1]

1 - LIFC, Univ. of Franche-Comté, France
2 - AlGorille Team, Loria, Univ. Henri Poincaré, Nancy, France

Contents

- Context and objectives
- Model
  - Notations
  - Assumptions
- Choice of load ratios
- Theoretical result
- Experimental evaluation
  - Context
  - Results
- Conclusion

Context and objectives

* Evolution of parallel systems
  * More and more complex:
    * Hierarchy and heterogeneity
    * Dynamicity: intermittent links

⟹ Strong demand for generic adaptive algorithms

⟹ Mutation from static/centralized LB algorithms to dynamic/decentralized ones

* The most suited are based on local exchanges but are either synchronous or assume a static net

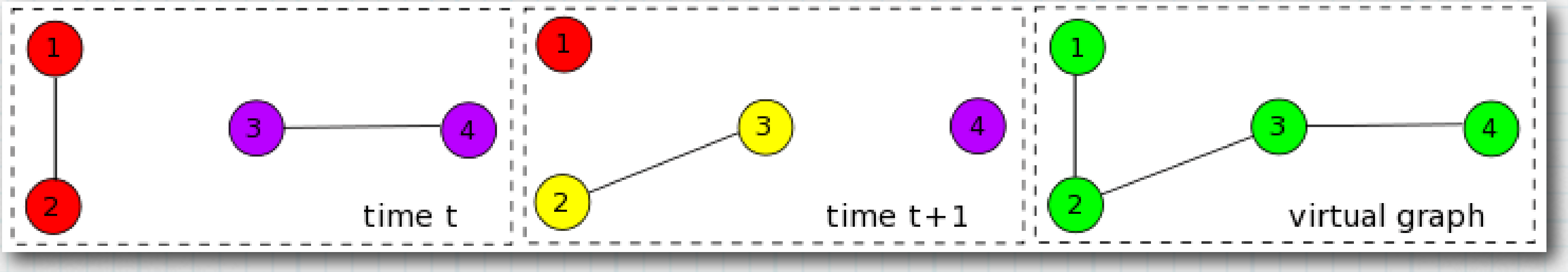
⟹ **Asynchronous Load Balancing!!**

Model

Notations	
n	number of processors
$P=\{1,...,n\}$	set of processors
$N_i(t)$	set of processors directly connected to i at time t
$x_i(t)$	load of processor i at time t
$x_j^i(t)$	evaluation on processor i at time t of the load of processor j
$s_{ij}(t)=\alpha_{ij}(t)(x_i(t)-x_j^i(t))$	amount of load sent by i to j at time t $0\leq\alpha_{ij}(t)\leq 1$, $\sum_{j=1}^{n}\alpha_{ij}(t)=1$, $\alpha_{ij}(t)=0$ if $j\notin N_i(t)$ or $x_i(t)\leq x_j^i(t)$

Model

- Assumptions:
  - Initial condition: $\sum_{i=1}^{n} x_i(0) = L$
  - Communication interruptions and delays are bounded
  - Jointly connected condition



  - Continuous representation of the loads
  - When two nodes are connected, the most loaded sends a part of its load to the other
  - Remaining load on the sender will not become smaller than those on the receivers

Load ratios

- Must ensure that the load on every node will converge towards L/n
- Let's consider $j^* = \min_{k \in Ni(t)} x_k^i(t)$
- We obtain the following constraints:

$$\sum_{k \in Ni(t) \setminus \{j^*\}} \alpha_{ik}(t)(x_i(t) - x_k^i(t)) \leq ß(x_i(t) - x_{j^*}^i(t))$$

where $0 \leq ß < 1$ is a real constant

and

$$\alpha_{ij^*}(t) = \frac{1}{2}\left(\frac{1 - \sum_{k \neq j^*} \alpha_{ik}(t)(x_i(t) - x_k^i(t))}{x_i(t) - x_{j^*}^i(t)}\right) \geq \frac{1 - ß}{2}$$

and

$$0 \leq \alpha_{ij}(t) \leq \frac{1}{2}\left(\frac{1 - \sum_{k \neq i} \alpha_{ik}(t)(x_i(t) - x_k^i(t))}{x_i(t) - x_j^i(t)}\right)$$

Theoretical result

Theorem

Under the previous assumptions, the asynchronous load balancing algorithm below converges to $x^* = 1/n \sum_{i=1}^{n} x_i(0)$

Algorithm

At each time step, each processor:

1. Compares its load to the loads of its connected neighbors
2. Determines the $\alpha_{ij}(t)$ and deduces the $s_{ij}(t)$
3. Sends those loads to the corresponding neighbors
4. Receives some load from its more loaded neighbors

Experimental evaluation

- Context:
  - Implementation in the SimGrid simulator
  - SimGrid:
    - Framework for testing algorithms in clusters/grids
    - Realistic computation and communication models
    - Reproducible and representative results
  - Asynchronous iterative algorithm:
    - Tasks are the elements of a domain discretization
    - Set of identical tasks with different numbers of iterations
    - Tasks can be migrated by our LB algorithm
      ⇒ one task can perform its iterations on different nodes

Local load distribution strategy

* Our theorem gives constraints on the load amounts to send to the neighbors but no precise values

  $\Rightarrow$ Several local load distribution strategies are possible

* The tested one:
  * $\alpha_{ij}(t) = 1/(|N_i(t)|+1)$ $\forall j \in N_i(t)$ s.t. $x_i(t) > x_j^i(t)$
  * Backpack-like load distribution to compute the $s_{ij}(t)$:
    * Not every less loaded neighbor actually receives load
    * Tests from the least to the most loaded neighbor
    * While the remaining local load stays larger than the load of the current neighbor plus its sent load ($s_{ij}(t)$)

Evaluation criteria

* Efficiency evaluation: two percentages
  * Overhead of our LB scheme according to a near optimal scheduling
    * Computation of the theoretical minimal makespan without taking into account the potential tasks migrations
      * Theoretical value, not always reachable
    * 10% means our LB is 10% slower than the optimal
  * Gain in execution time with our LB scheme relatively to the non-balanced version
    * 10% means our LB saves 10% of initial execution time

Experimental contexts

Cluster	
Size	10 and 50 machines
Powers	identical or different (ratio 10)
Links	homogeneous
Initial distribution of tasks	
or	All on a single node
	Evenly distributed over the processors
Communications	
or	Always active
	Intermittent
Tasks	
Number	10000
Data size	80 bytes per task
Iterations	random in [100, 500]
Flops	1600 per iteration

Linear topology

* Most difficult case due to the longer diffusion time

Size	Initial tasks distribution	Homogeneous procs		Heterogeneous procs		Measures
		10	50	10	50	
Constant links	All tasks on one node	31.38 *86.86*	387.82 *90.24*	34.96 *92.09*	367.50 *83.90*	% overhead % gain
	Even distribution	0.44 *0.97*	2.33 *3.13*	16.25 *80.64*	46.26 *75.31*	% overhead % gain
Intermittent links	All tasks on one node	55.58 *84.44*	967.35 *78.65*	146.73 *85.54*	832.17 *67.89*	% overhead % gain
	Even distribution	0.48 *0.93*	3.18 *2.33*	52.78 *74.56*	99.89 *66.26*	% overhead % gain

Ring topology

* Better suited context due to the smaller diameter

Size	Initial tasks distribution	Homogeneous procs		Heterogeneous procs		Measures
		10	50	10	50	
Constant links	All tasks on one node	11.55 *88.85*	292.48 *92.15*	23.43 *92.76*	370.14 *83.80*	% overhead % gain
	Even distribution	0.26 *1.15*	2.08 *3.37*	2.78 *82.89*	44.39 *75.63*	% overhead % gain
Intermittent links	All tasks on one node	23.75 *87.63*	1187.76 *74.24*	127.99 *86.64*	1116.72 *58.09*	% overhead % gain
	Even distribution	0.54 *0.87*	3.45 *2.07*	34.94 *77.53*	80.62 *69.51*	% overhead % gain

Complete graph topology

* Most favorable context due to the smallest diameter
* Fair results due to the load distribution strategy

Size	Initial tasks distribution	Homogeneous procs		Heterogeneous procs		Measures
		10	50	10	50	
Constant links	All tasks on one node	6.12 *89.39*	811.01 *81.78*	15.24 *93.25*	791.51 *69.29*	% overhead % gain
	Even distribution	0.4 *1.01*	7.45 *-1.72*	2.8 *82.89*	108.62 *64.79*	% overhead % gain
Intermittent links	All tasks on one node	28.11 *87.19*	4101.52 *15.97*	46.96 *91.39*	1085.86 *59.15*	% overhead % gain
	Even distribution	0.31 *1.09*	6.74 *-1.04*	7.93 *82.03*	331.93 *27.09*	% overhead % gain

Complete graph topology

- Most favorable context due to the smallest diameter
- Fair results due to the load distribution strategy

Size	Initial tasks distribution	Homogeneous procs		Heterogeneous procs		Measures
		10	50	10	50	
Constant links	All tasks on one node	6.12 *89.39*	811.01 *81.78*	15.24 *93.25*	791.51 *69.29*	% overhead % gain
	Even distribution	0.4 *1.01*	4.67 *0.92*	2.8 *82.89*	108.62 *64.79*	% overhead % gain
Intermittent links	All tasks on one node	28.11 *87.19*	4101.52 *15.97*	46.96 *91.39*	1085.86 *59.15*	% overhead % gain
	Even distribution	0.31 *1.09*	3.31 *2.21*	7.93 *82.03*	331.93 *27.09*	% overhead % gain

- A slightly different strategy avoids the losses (in red)

Conclusion

* A decentralized LB scheme for use in dynamic networks has been presented
* It is generic and can be used in conjunction with many computing algorithms
* Its convergence has been proved for constant global load
  * But it should also be good for occasional load variations
* Experimental results confirm the interest of the method
  * No sensible overhead in already balanced cases
  * Good gains in the other cases
* Optimal choice of the local distribution strategy must be further investigated

A study of the (auto-)tuning of the inner parameters (ß and α_{ij}) and the implementation for use with a real application are the next steps!!