Robust and Tuneable Family of Gossiping Algorithms

Vincenzo De Florio

http://www.pats.ua.ac.be/vincenzo.deflorio

The 20th Euromicro Int.l Conference on Parallel, Distributed and Network-Based Computing
Garching, February 16, 2012
Structure

- Introduction – main key words
- Definitions, goals, and assumptions
- Algorithms and tuning algorithms
- Conclusions and lessons learned
Robust and Tuneable Family of Gossiping Algorithms:
three key words

Gossiping

Tuneable

Robust
Gossiping

- All-to-all pairwise IPC
- Every member of a set communicates a private value to all other members

- Useful e.g. in restoring organs, distributed consensus, interactive consistency...
Tuneable

- An example of a tuneable algorithm
  - An **application layer "knob"** to select certain behaviours rather than others
    - Explicit knob:
      - one *we are aware of*
      - and *know how to steer*
    - (As opposed to a hidden knob: one that we don't know it exists nor how to use!)
A **robust** knob: one that allows
- persistence of certain *behaviours* despite *changes* in the *context*
- to match fundamental assumptions regarding the environment in which the system operates [Jen04]
- An in particular, to match *dynamically changing assumptions*: robustness throughout system evolution

→ **Resilience**
Definitions

• Contextual parallelism (CP): the physical parallelism available in the system and the network
• Algorithmic parallelism (AP): the number of independent "threads" of activity expressed by an algorithm

• Algorithmic undershooting: $\text{AP} < \text{CP}$
  o The algorithm under-utilizes the available resources (sub-optimal behaviour)
• Algorithmic overshooting: $\text{AP} > \text{CP}$
  o The algorithm requests more resources than physically available
  o Unnecessary burden for system/network layers: request queues overloading, flooding, collisions, ...
    o One needs complexity just to deal with the overhead one introduces!
Main design goal

- Dynamically robust tuning:
  - Optimal match between AP and CP over time
  - Dynamic avoidance of **shortcoming** or **excess** of algorithmic parallelism
Assumptions

- \( N + 1 \) processors (\( N > 1 \))
- Full-duplex point-to-point communication lines
- Communication: synchronous and blocking
- Processors are uniquely identified by integer labels \( \in I_N = \{0, \ldots, N\} \)
- Each processor \( p_i \) owns some local data \( v_i \)
- Each processor requires the other processors’ local data and then executes some algorithm (e.g. voting)
- Events occur by discrete time steps
State templates

- WR state. A process is waiting for the arrival of a message from some processor.
  - Lasts zero or more time steps
- R state. Process $i$ receiving at time $t$ a message from process $j$ is said to be in state $R_{ij}$. This is represented as $i R^t j$
  - Lasts one step
- WS state. Process $i$ waiting to send process $j$ its message is said to be in state $WS_{ij}$.
  - Lasts zero or more time steps
- S state. Process $i$ is at time $t$ in state $S_{ij}$ when it is sending a message to process $j$. This is represented as $i S^t j$
  - Lasts one time step
Index permutation

- Process $i$ owns an index permutation, i.e. an array whose members are permutations of
  $$\{0, \ldots, i-1, i+1, \ldots, N\}$$

- $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$ is used to represent the index permutation
Algorithm – first formulation

Process \( i \) executes:

- \( i \) times

\[ \text{Index is } P \]

\[ N \text{ times} \]

\[ N - i \text{ times} \]

Algorithm – v1.0 (1/4)
Properties

• Whatever the index permutation, the algorithm does work
• In [DeBl06] we analyzed the efficiency of the algorithm when $\mathcal{P}$ takes a given structure
• In particular,
  • When $\mathcal{P}$ is the identity permutation :
    $\mathcal{P} = \{ 0, ..., i - 1, i + 1, ..., N \}$
  • When $\mathcal{P}$ is the pipelined permutation :
    $\mathcal{P} = \{ i + 1, ..., N, 0, ..., i - 1 \}$
Identity permutation, $N = 5$

<table>
<thead>
<tr>
<th>id</th>
<th>step</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>$S_1$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_1$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
<td>$R_5$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>$R_0$</td>
<td></td>
<td>$S_0$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>$R_0$</td>
<td></td>
<td>$S_0$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>$R_0$</td>
<td></td>
<td>$S_0$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$R_0$</td>
<td></td>
<td>$S_0$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$R_0$</td>
<td></td>
<td>$S_0$</td>
<td>$S_2$</td>
<td>$S_3$</td>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$R_2$</td>
<td></td>
<td>$R_3$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
<td></td>
<td>$R_4$</td>
</tr>
</tbody>
</table>

- Table = transcript of states
  - e.g. Proc. 3 at step 8 receives from proc 1 = $3 R^8 1$
  - Arrow-to-the-left = WR; arrow-to-the-right = WS

- AP = about 2.31
- Asymptotic value of $AP(N) = 8/3$
Pipelined permutation, $N = 9$

- $AP = 6$
- Asymptotic value of $AP(N) = 2(N+1)/3$
- Multiple consecutive sessions sustain the steady state
Algorithm, take 2

\[ N \times i \text{ times} \]

Select a "lane"

\[ N \text{ times} \]

\[ N - i \text{ times} \]
Algorithm, v2.0

• In what follows we consider just two types of “lanes”: either with identity or with pipelined permutations
• We assign $H\%$ to pipelined and $(100-H)\%$ to identity
  • ($H$ for “Hybrid”)
• Results:
  • $\text{AP(identity)} \leq \text{AP}(H) \leq \text{AP(pipelined)}$
  • $\text{AP}(H)$ increases with $H$
Algorithm, v2.0

Algorithm – v2.0 (3/3)
Observations

→ This paves the way to context aware adaptation:

A MAPE adaptation loop:

M: Estimate $CP(now)$
A: assess how $AP(now)$ matches $CP(now)$:
  Case of overshooting? Case of undershooting?
P: If *-shooting, select $H\%$ so as to make
  $AP(now')$ “closer” to $CP(now)$
E: use Algorithm v2.0 with selected $H\%$

so as to guarantee our design goals
(dynamic robust tuning)
Observation

• Instead of running Alg. v2.0 to compute AP values, we may use offline-computed values:
  \[ M(N, h) = AP \]

• A look-up table storing the algorithmic parallelism corresponding to \( CP(t) = N \) and \( H\% = h \)
Adaptation algorithms

- Algorithm “Tune AP after CP”, v.1
  Input: CP(t), N, M( (N, h) → AP )
  Output: H% best matching CP(now())

begin
  \( cp \leftarrow \text{sense}(CP(\text{now}())) \) // \( cp \) is now the current
  // physical parallelism
  \( H \leftarrow \min\{h \mid M(N,h) \geq cp\} \) // \( H \): sample
    // corresponding to
    // the supremum

  \text{return} H
end .

- Observation: A “growing” look-up table would allow to
  keep track of past decisions
    \( \text{growMap} (M, \text{new} ( (N, h) \rightarrow AP )) \)
Adaptation algorithms

• Algorithm "Tune AP after CP", v.2
  Input: \( CP(t), N, M( (N, h) \rightarrow AP ) \)
  Output: \( H\% \) best matching \( CP(now()) \)

begin
  \( cp \leftarrow \text{sense}(CP(now())) \)
  \( sup \leftarrow \text{min} \{ h \mid M(N,h) \geq cp \} \)
  if \( M(N,sup) == cp \) then return \( sup \) end-if
  // \( M(N,sup) > cp \)
  \( inf \leftarrow \text{max} \{ h \mid M(N,h) < sup \} \)
  newH \leftarrow ( M(N,sup) - M(N,inf) ) / 2
  \( ap \leftarrow \text{compute}(newH) \) // executes a run
  // and returns the AP
  \( \text{growMap} (M, (N,newH) \rightarrow ap ) \)
  return newH
end .

Robust tuning (2/4)
Run of adaptation algorithm v2, $N=200$
A run of algorithm v2

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>2.653465</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>134</td>
<td></td>
<td>50</td>
<td>5.170751</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>75</td>
<td>8.621984</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>81.5</td>
<td>11.018227</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>81.5</td>
<td>11.018227</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2.653465</td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td>100</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>84.5</td>
<td>12.753807</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>84.5</td>
<td>12.753807</td>
</tr>
<tr>
<td></td>
<td>85</td>
<td>13.105134</td>
<td></td>
<td>86</td>
<td>13.876424</td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>84.5</td>
<td>12.753807</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2.653465</td>
<td></td>
<td>84.5</td>
<td>12.753807</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5.170751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.5</td>
<td>15.241706</td>
<td></td>
<td>100</td>
<td>134</td>
</tr>
</tbody>
</table>

Robust tuning (4/4)
Conclusions & Lessons Learned

• An application-layer “knob” to tune algorithmic parallelism
• A knob that allows us to achieve robustness throughout system evolution (ability to match dynamically changing assumptions) = resilience
Conclusions & Lessons Learned

• When performing e.g. cross-layer adaptation, one deals with many intertwined VMs
  • Application layer, application server layer, OS, the network layers, the HW...
→ Stigmergy everywhere!
• These VMs may appear to the adaptation engineer as
  • White boxes: known robust knobs we are aware of
  • Grey boxes: known knobs we are partially aware of
  • Black boxes: hidden knobs
• Excluding the application layer (the algorithm) **locks in** to inefficiency or non-robustness
→ We need to expose the algorithmic knobs!
→ Sort of an “End-to-end Argument” in system evolution [SaRC84]
Main sources


Systems robust throughout their evolution:

More info: http://www.igi-global.com/ijaras