Pyrros+: Automatic parallelization and performance prediction tool

• Apostolos Gerasoulis
• Rutgers University, USA
  www.cs.rutgers.edu/~gerasoul
Colaborators

- Tao Yang, UCSB
- Michel Cosnard, Michel Loi, France
- Jia Jiao, Bell Labs- Lucent
- David Rhodes, US Army CECOM/RDEC.
- Rutgers Students:
  - Chengli Wang
  - Yingfang Lu
  - Baohua Wu
Supported

- DARPA (Hpcd, Prediction)
- NSF-INRIA (travel)
- SUN Microsystems (22 Ultraspark)
- HP (40 Kayak)
- Rutgers University (HPC Lab)
- Rutgers University (5 New system hires)
Research area

• **Static & Dynamic Scheduling**
  - Program information at compile and run time, e.g. computation and communication
  - Processor information at compile and run time, e.g. processor load

• **Scheduling Tools**
  - PYRROS and D-PYRROS: Static and dynamic scheduling systems
  - Pyrros+: Automatic parallelization and scheduling
Goals

• Predict Program Performance: Pyrros+
• Use Prediction to improve the performance of parallel programs:

Examples:
Harmonic Balance (Rhodes),
Ship Design (SAIC),
Linear Algebra (Tao Yang),
Nbody Appl. (Gerasoulis)
Program Prediction Range

- ANALYTIC
- STATISTICAL

\[ T_p = n^3 w/p + n^2 b \]

Isoeficiency Hint

- MORE EXPENSIVE -----HIGHER ACCURACY
- LESS EXPENSIVE -----LESS ACCURACY

EXACT

EMULATORS

PYRROS+

LINPACK PERFORMANCE
Pyrros+

- **INPUT**: Sequential program with task annotations
- **OUTPUT**: Parallel Program using PYRROS scheduler
- **PREDICTS** performance using Program information, processor and network speed estimation.
- **INOVATION**: First task based automatic parallelization and scheduling system
Pyrros+ Technology

- Omega Test for Fine grain analysis.
- Macrodataflow model for coarse grain analysis
- Pyrros static scheduling
- Pyrros code generation in MPI.
- Graphical user interface
Pyrros+ architecture

Sequential program

Initial Partitioning

Task Graph

Final Partitioning

Partitioned Graph
Partitioned Data

Scheduling

Code Generation Optimization

Parallel Code

Architecture Parameters

Granularity Control
Pyrros+ Example

- param n
- real a(n, n)
- for j = 1 to n do
- for i = 1 to n do
  - task
  - a(i, j) = a(i, j-1) + 1
  - endtask
- endfor
- endfor
Pyrros+ Interface

- A loop example at statement level partitioning
- Partitioning at interior loop
- Loop interchange to create parallelism at coarser grain
Pyrros+ estimation model

- **Computation**: number of operations times the cost per operation
- **Communication**: Linear model- $a + \text{size} \times b$, where $a$ is processor overhead and $b$ is transmission rate
- **Scheduling**: Macrodatflow and Pyrros scheduling.
## Pyrros Performance

- **Gauss Jordan algorithm**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Predicted time</th>
<th>Actual time</th>
<th>Seq. time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SP-2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P=2</td>
<td>19.37(s)</td>
<td>20.75(s)</td>
<td>23</td>
</tr>
<tr>
<td>P=4</td>
<td>11.86</td>
<td>14.96</td>
<td></td>
</tr>
<tr>
<td>MYRINET &amp; ULTRA-1 167MHZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P=2</td>
<td>24.74</td>
<td>23.50</td>
<td>53</td>
</tr>
<tr>
<td>P=4</td>
<td>17.32</td>
<td>15.75</td>
<td></td>
</tr>
<tr>
<td>NCUBE-2S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P=2</td>
<td>625(s) (*)</td>
<td>581</td>
<td>1152(*)</td>
</tr>
<tr>
<td>P=4</td>
<td>328</td>
<td>336</td>
<td></td>
</tr>
<tr>
<td>P=8</td>
<td>184</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

- **PROBLEM SIZE**
  - n=1000, N=50, r=10

(*) Estimated
Pyrros+ Performance

- Gauss Elimination algorithm

<table>
<thead>
<tr>
<th>Machines</th>
<th>Actual time</th>
<th>Speedup</th>
<th>nproc</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 SUN ULTRA</td>
<td>31.7(sec)</td>
<td>2 (est)</td>
<td>2</td>
</tr>
<tr>
<td>167Mhz + 300Mhz.</td>
<td>15</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Gbit Myrinet</td>
<td>8.7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>MPI MPICH</td>
<td>6.7</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>8.6</td>
<td>7</td>
<td>16</td>
</tr>
</tbody>
</table>

PROBLEM SIZE: n=1200, N=30, r=40
Can Pyrros+ Predict Performance?

- Gauss Elimination algorithm

<table>
<thead>
<tr>
<th>Machines</th>
<th>Predicted</th>
<th>Actual</th>
<th>nproc</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 SUN ULTRA 167Mhz</td>
<td>90(sec)</td>
<td>88.76 (est)</td>
<td>1</td>
</tr>
<tr>
<td>100 Mbit ethernet</td>
<td>50.5</td>
<td>50</td>
<td>2(**)</td>
</tr>
<tr>
<td>MPI MPICH</td>
<td>38.5</td>
<td>37</td>
<td>3</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>32.5</td>
<td>30.6</td>
<td>4</td>
</tr>
<tr>
<td>MPE OF</td>
<td>27.6</td>
<td>26.7</td>
<td>5</td>
</tr>
<tr>
<td>????????????????????</td>
<td>22.3</td>
<td>19.7</td>
<td>8</td>
</tr>
<tr>
<td>????????????????????</td>
<td>16</td>
<td>26.6</td>
<td>12</td>
</tr>
</tbody>
</table>

PROBLEM SIZE
n=1600, N=40, r=40

(**) Parameters where estimated from nproc=2
Comm. Load, OS overhead was set =0
David Rhodes Application

• Pyrros+ : Going Beyond the automatic parallelization restrictions!!!
• Problem: Parallelize the Harmonic Balance Simulation equation
• Goal: Achieve high speedups; e.g. 20 frequencies run in parallel gives 20 speedup; can we go beyond 20 speedup?
David Rhodes Application

• Each frequency is a Tree task graph!
• David Estimated the weighted task graph (169 nodes) for the ARMY CRAY T3E
• Pyrros+ Predicted a maximum spedup of 69 using 120 processors! It also proposed a scheduling to achieve this.
• WORK IN PROGRESS
Why Access Pattern is important?

- **Sequential Gauss Jordan**

<table>
<thead>
<tr>
<th>Machines</th>
<th>row major</th>
<th>column major</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS/6000(SP2)</td>
<td>514(sec)</td>
<td>23.3(s)</td>
</tr>
<tr>
<td>Ultraspark 140</td>
<td>175</td>
<td>53.9</td>
</tr>
<tr>
<td>Ncube2S</td>
<td>924</td>
<td>913</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROBLEM SIZE</th>
<th>n=1000</th>
<th></th>
</tr>
</thead>
</table>
What improvements are needed?

- Parameter estimation needs to be expanded to include processor and communication load statistics
- Data access patterns need to be added to the operation count statistics
- Alternative Scheduling algorithms need to be tested
Pyrros vs. Pyrros+

\[ n=1000, \ N=\frac{1000}{10}=100, \ r=10 \]
D-PYRROS

- **Innovations**
  - Incremental scheduling
  - Run time program computation and communication estimation for scheduling.
  - Local scheduling to lower scheduling overhead.
  - Local or global clustering (DSC) at run time to reduce high communication cost.
  - Re-scheduling (re-mapping) of computation based on run time multiprocessor performance deterioration.
  - Excellent performance for "slowly changing dynamic problems".

- **New in 98: MPI parallel code.**
2D vortex dynamics

LOCAL computation tree

150 steps
D-PYRROS Performance

![Graph showing D-PYRROS Performance]
Plans 1997-2000

- **1997-1998: Pyrros+ prototype.**
  - Integrated GUI and Automatic Parallelization, Integrated Pyrros and MPI. First experimental results.

- **1998-1999: D-Pyrros and Pyrros+**
  - Incorporate dynamic scheduling. Incorporate Access Patterns in computation estimation, Incorporate Computation and communication load. OUT of CORE application emulators.

- **1999-2000 Irregular Pyrros+**
  - Integrate RAPID with Pyrros+, Symbolic scheduling, Large scale irregular applications.