Optimal Subset Mapping And Convergence Evaluation of Mapping Algorithms for Distributing Task Graphs on Multiprocessor SoC

Heikki Orsila, Erno Salminen, Marko Hännikäinen, Timo D. Hämäläinen

heikki.orsila, erno.salminen, marko.hannikainen, timo.d.hamalainen@tut.fi

Institute of Digital and Computer Systems
Tampere University of Technology, Finland
Presentation Outline

- Introduction
- Experiment
- Algorithms
- Comparison of algorithms
- Conclusions
- References

+ all the algorithms, graphs and pictures in the end
Introduction (1/2)

- Automatic distribution of *process networks* onto a multiprocessor system while satisfying some specific criteria.
- Assume $N$ tasks in the process network, and $M$ processing elements (PEs) in the multiprocessor system.
- Define *mapping* as one possible placement of $N$ tasks to $M$ processing elements (task $i$ goes to PE $x$, task $j$ goes to PE $y$, ...).

![Task graph and mapping diagram](image-url)
The problem is to minimize a cost function
  • Minimizing the cost function often means maximizing performance or optimizing some other property
  • NP problem $\rightsquigarrow$ true minimum is not generally achieved or known, but maybe the result is good enough

Also, try to minimize optimization time
  $\rightsquigarrow$ trade-off between a good result and short optimization time
  • This is important in exploration of large design space
  • Optimum solution for distribution varies with the architecture
Contributions

• A new mapping algorithm called *Optimal Subset Mapping* (OSM)
  • OSM sacrifices result goodness to decrease optimization time
• Comparison of mapping algorithms with respect to result goodness, optimization time and converge
• Supporting evidence for our simulated annealing parametrization method presented in [2][3]
• These methods are suitable for both shared and distributed memory systems
Experiment

• Compare 6 algorithms
• 10 random graphs, $N = 300$ nodes
• Simulation run 10 times independently, results averaged
• $M = 2, 4$ and $8$ processing elements connected with a shared bus
• Measure speedup with respect to a single processor system
Algorithms (1/3)

• Use algorithms that have *reasonable* polynomial optimization time upper-bounds with respect to number of tasks $N$ and processing elements $M$

• Upper-bounds for mappings tried for algorithms:
  - Optimal subset mapping (OSM): $O\left(\frac{N^2 M}{\log N + \log M}\right)$
  - Our simulated annealing variant (SA+AT):
    $O\left(NM \log \frac{T_0}{T_f}\right)$
  - Group Migration (GM): $O(N^2 M)$
  - Random mapping: fixed number of iterations (only used as a reference)
Algorithms (2/3)

- Group Migration (GM), also known as Kernighan-Lin graph partitioning algorithm
  - deterministic
  - greedy, may get stuck into local minima
- SA+AT is our version of the simulated annealing algorithm [3]
  - Stochastic and non-greedy
  - Automatic temperature (AT) scale is determined from the graph
  - Transition probabilities are normalized for efficient optimization
  - Fully automated parameter selection \( \rightsquigarrow \) requires no manual tuning of parameters
- The hybrid algorithm [4] is a combination: result of SA is a starting point for GM
Algorithms (3/3)

- OSM is the Optimal Subset Mapping algorithm
  - A divide and conquer algorithm. Solves a subset of the problem optimally, but does not guarantee global optimum
  - Picks a subset of tasks and brute-forces an optimal mapping for the subset, and then picks another subset and optimizes that
  - The subset size is increased and decreased continuously when and if there is potential for optimization
  - When increasing the subset size does not improve the result anymore, the algorithm terminates
The following figure shows convergence for 8 processing elements for each algorithm. The X-axis is the number of mappings tried (logarithmic scale). The Y-axis is the average speedup (1.0 means no speedup) over all graphs.
Comparison of Algorithms (2/3)

Following table shows speedups and convergence rate for each algorithm:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Speedup</th>
<th>Speedup / Iterations</th>
<th>Convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>1.76</td>
<td>1.0 (reference level)</td>
<td>Too long</td>
</tr>
<tr>
<td>OSM</td>
<td>3.25</td>
<td>6.11</td>
<td>Fast</td>
</tr>
<tr>
<td>GM</td>
<td>3.38</td>
<td>1.21</td>
<td>Slow</td>
</tr>
<tr>
<td>SA+AT</td>
<td>3.65</td>
<td>2.58</td>
<td>Fast</td>
</tr>
<tr>
<td>Hybrid</td>
<td>3.69</td>
<td>0.20</td>
<td>Slow</td>
</tr>
</tbody>
</table>
Comparison of Algorithms (3/3)

- Random mapping shows the base-level for optimization.
- OSM is most suited for comparing architectures and systems rapidly, but does not yield good speedup.
- GM is not suitable for architecture exploration as it is slow and does not yield good speedup.
- SA+AT is strong both in convergence speed and speedup.  $\rightarrow$ this is currently our algorithm of choice.
- Hybrid algorithm yields the best speedup, but it is slow.

Future directions:

- Combine features of each algorithm. For example, start with OSM, and after rapid initial convergence, switch to SA+AT.
- Try genetic algorithms. Problem: hard to select proper parameters.
Discussion

- Almost all papers on task distribution that use Simulated Annealing leave some parameters undocumented
  - Hard to learn about Simulated Annealing even if there are lots of papers that use it
  - We were motivated to document parameters of Simulated Annealing properly [2] [3]
- We use random graphs to avoid application bias in performance
- Static acyclic graphs have very well known scheduling properties, and hence, differences in results are due to mapping algorithms
- Group migration is highly sensitive to initial values, but other algorithms are not
Conclusions

• This paper demonstrates convergence properties of several algorithms
• This paper demonstrates that automatic parameter selection for simulated annealing can be effective
• SA+AT algorithm converges rapidly, but still yields very good results
• The new OSM algorithm converges very rapidly, but does not yield very good results. It is still suitable for comparing architecture and system alternatives in architecture exploration.


References (2/3)


Optimal Subset Mapping Pseudo-code

```
OPTIMAL_SUBSET_MAPPING(S)
1  $S_{best} \leftarrow S$
2  $C_{best} \leftarrow \text{Cost}(S)$
3  $X \leftarrow 2$
4  for $R \leftarrow 1$ to $\infty$
5    do $C_{old_{best}} \leftarrow C_{best}$
6      $S \leftarrow S_{best}$
7      $\text{Subset} \leftarrow \text{Pick Random Subset}(S, X)$
8      for all possible mappings $S_{sub}$ in $\text{Subset}$
9        do $S \leftarrow \text{Apply Mapping}(S, S_{sub})$
10       $C \leftarrow \text{Cost}(S)$
11       if $C < C_{best}$
12          then $S_{best} \leftarrow S$
13          $C_{best} \leftarrow C$
14       if $\text{modulo}(R, R_{max}) = 0$
15         then if $C_{best} = C_{old_{best}}$
16            then if $X = X_{max}$
17              then break
18            else $X \leftarrow X + 1$
19         else $X \leftarrow X - 1$
20       $X \leftarrow \text{Max}(X_{min}, X)$
21       $X \leftarrow \text{Min}(X_{max}, X)$
22     return $S_{best}$
```
# Application and architecture parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># graphs</td>
<td>10</td>
</tr>
<tr>
<td># tasks per graph (N)</td>
<td>302</td>
</tr>
<tr>
<td># edges per graph</td>
<td>(1594, 5231, 8703)</td>
</tr>
<tr>
<td>comp time per task [us]</td>
<td>(3.2, 5.1, 7.0)</td>
</tr>
<tr>
<td>comm vol per task [byte]</td>
<td>(26, 1111, 3679)</td>
</tr>
<tr>
<td>comm/comp -ratio [Mbyte/s]</td>
<td>(8, 218, 526)</td>
</tr>
<tr>
<td>max theor. parallelism [no unit]</td>
<td>(4.3, 7.9, 12.8)</td>
</tr>
</tbody>
</table>

| Task graphs | |
| # PEs \(M\) | 2, 4, 8 |
| PE freq [MHz] | 50 |
| Bus Freq [MHz] | \(10, 20, 40\) |
| Bus width [bits] | 32 |
| Bus bandwidth [Mb/s] | \(320, 640, 1280\) |
| Bus arb. latency [cycles/send] | 8 |

| HW Platform | |
| # runs per graph per alg | \(10\) |
| algorithms | 6 |
| determ, non-greedy | 1: OSM |
| determ, greedy | 1: GM |
| stoch., non-greedy | 4: SA, SA+AT, hybrid, random |
| stoch, greedy | - |

Notes:

\(1\) = min, avg, max
\(2\) = values for 2,4,8 PEs, respectively
\(3\) = only 1 run for GM
## Optimization parameters

<table>
<thead>
<tr>
<th>Alg.</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># iter per $T$, $(L = N \cdot (M-1))$</td>
<td>602, 1208, 2416</td>
</tr>
<tr>
<td></td>
<td># temperature levels</td>
<td>181</td>
</tr>
<tr>
<td></td>
<td># temperature scaling</td>
<td>$q = 0.95$</td>
</tr>
<tr>
<td></td>
<td>range of $T$ (SA and hybrid)</td>
<td>$T_0 = 1.0$, $T_f = 0.0001$</td>
</tr>
<tr>
<td></td>
<td>range of $T$ (SA+AT)</td>
<td>$T$ range coefficient $k = 2$</td>
</tr>
<tr>
<td></td>
<td>annealing schedule $(T_0, i)$</td>
<td>$T_0 \cdot q^{\text{floor}(i/L)}$</td>
</tr>
<tr>
<td></td>
<td>move heuristic</td>
<td>move 1 random task</td>
</tr>
<tr>
<td></td>
<td>acceptance function</td>
<td>$(1 + \exp(\Delta C / (0.5 \cdot C_0 \cdot T)))^{-1}$</td>
</tr>
<tr>
<td></td>
<td>end condition</td>
<td>$T = T_f$ AND $L$ rejected moves</td>
</tr>
<tr>
<td>Rand</td>
<td># max iterations</td>
<td>262, 144</td>
</tr>
<tr>
<td>GM</td>
<td>no params needed</td>
<td>-</td>
</tr>
<tr>
<td>OSM</td>
<td>coefficient $c$</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>exponent $c_N$</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>exponent $c_M$</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>subset size $X$ [#tasks]</td>
<td>9, 5, 3</td>
</tr>
<tr>
<td></td>
<td># iterations per subset</td>
<td>512, 1024, 512</td>
</tr>
</tbody>
</table>

**Notes:**

(1) = values for 2, 4, 8 PEs, respectively

(2) = $T_0$ and $T_f$ computed automatically in SA+AT
### Rounds and mapping iterations for OSM

<table>
<thead>
<tr>
<th>PEs</th>
<th>rounds (min, avg, max)</th>
<th>Thousands of iterations (min, avg, max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>271, 380, 611</td>
<td>34.1, 37.2, 73.6</td>
</tr>
<tr>
<td>4</td>
<td>239, 469, 899</td>
<td>80.6, 115.4, 259.1</td>
</tr>
<tr>
<td>8</td>
<td>199, 428, 1099</td>
<td>57.1, 88.8, 293.9</td>
</tr>
</tbody>
</table>
Best gain divided by the number of iterations

![Bar chart showing the gain divided by the number of iterations for different mapping algorithms.](chart.png)
OSM progress plotted for each graph
SA+AT progress plotted for each graph
GM progress plotted for each graph