Stochastic Partitioning of Process Networks

O. Stan[1,2], R. Sirdey[1], J. Carlier[2] and D. Nace[2]

[1] Commissariat à l'Énergie Atomique
[2] Université de Technologie de Compiègne

Séminaire PGMO
Ecole Polytechnique
16 Octobre 2012

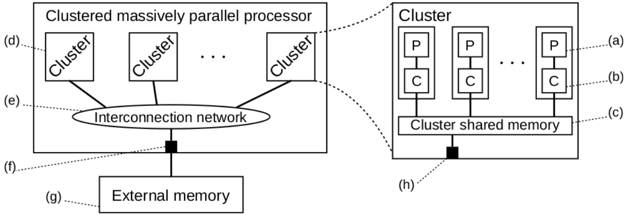
Contents

Context

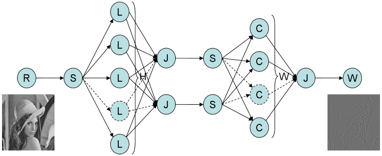
- Computation intensive embedded systems
  - New Moore law :
    - The number of ~~transistors~~ cores doubles every 2 years
  - New generation of microprocessor architectures : embedded manycores
    - Massively (100+) multi-core systems on-chip
- Difficulties in developing applications for these architectures
  - Running correctly large parallel programs
  - Efficiently exploiting the parallelism
  - Performance constraints and dependability requirements
  - Limited resources
- Need of theoretical and practical advances in :
  - Programming models and languages
  - Innovative compilation technologies
  - Suitable operations research techniques

Manycore architectures

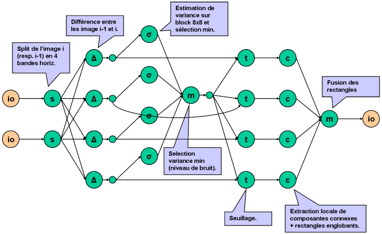
- Manycore system : A parallel computing system integrating a number of processing cores, a mix of shared and local memory, distributed global memory or multilevel cache hierarchy and an interconnection network-on-chip (NoC).
- Example : Kalray MPPA (Massively Parallel Processor Array)
  - A clustered massively (200+) multicore architecture
  - The clusters are MIMD (Multiple Instructions Multiple Data) parallel computing systems with several cores and a shared memory, connected via an on-chip asynchronous packet network with a 2D tore topology



- Dataflow models
  - A network of agents/tasks communicating through unidirectional FIFO channels
  - Exclusively data-driven synchronization
- ΣC Language
  - Adapted to a wide range of embedded applications
  - High-level : no mention of the memory hierarchy or chip layout
  - Explicit expression of parallelism
  - Extension to C



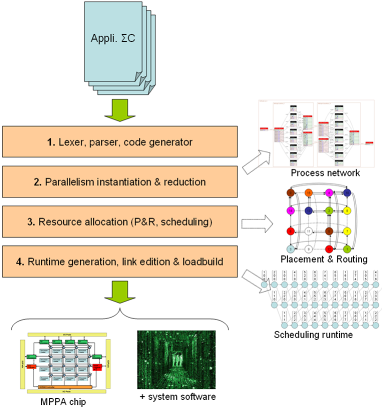
Laplacian computation for an image.



Target tracking pipeline application.

Compilation process

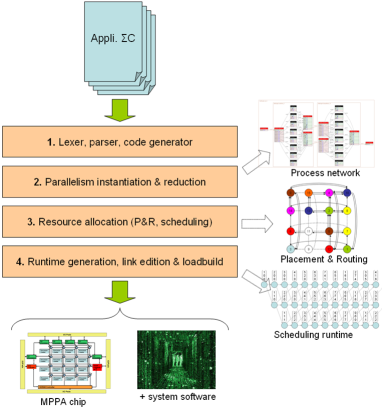
- Lexical, syntactic, semantic etc. analysis.
- Code generation.
- Construction of the process networks.
- Composition of data access patterns, Parallelism reduction.
- Buffer dimensioning.
- Construction of a partial ordering of tasks occurrences.
- Partitioning/Placement/Routing.
- Generation of runtime tables.
- Loadbuild.
- Execution.
- And. . . Iterative compilation.



Compilation of a ΣC application.

Compilation process

- Lexical, syntactic, semantic etc. analysis.
- Code generation.
- Construction of the process networks.
- Composition of data access patterns, Parallelism reduction.
- Buffer dimensioning.
- Construction of a partial ordering of tasks occurrences.
- **Partitioning/Placement/Routing.**
- Generation of runtime tables.
- Loadbuild.
- Execution.
- And. . . Iterative compilation.



Compilation of a ΣC application.

Placement & routing

- Partitioning :
  - Group together communicating tasks, under multidimensional capacity constraints.
  - Economic function : NoC bandwidth.
  - Class of problem : node capacitated graph partitioning.
  - Order of mag. : thousands of tasks (maybe much more in fact...), tens of partitions.
- Placing :
  - Assignment of groups of tasks to nearby architectural elements.
  - Economic function : latency.
  - Class of problem : quadratic assignment.
  - Order of mag. : tens of groups, tens of nodes.
- Routing :
  - Computation of data routes accross the NoC.
  - Economic function : latency & link load.
  - Class of problem : (constrained) multi-flow.
  - Order of mag. : tens of flows, tens of nodes.
- NP-hard underlying discrete optimization problems.

Approaches

- Beginning of the development cycle :
  - Sequential approach : partition then place then route (using fast–few seconds–approximate algorithms).
    - **GRASP for partioning**.
    - Simulated annealing for placing.
    - Integer programming for routing.
- Towards the end of the dev. cycle :
  - Global approach : partition and place and route in one go.
    - But the problem is much more complex and intrinsically multi-criterion, thus its resolution is much more computationally involved.
    - Master (partioning) and slave (routing) approach (leveraging on the previous algorithms) along with parallelization.
    - Few tens of minutes on a 50 cores parallel computer.

Partitioning problem statement

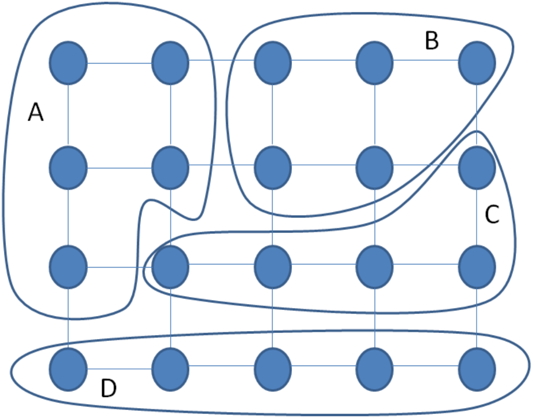
- Partitioning of Process Networks
  - Dataflow application graphs
  - Group together tasks which communicate more, under resource constraints, in order to optimize a network bandwidth criterion.
- Let $G = (V, A)$ be a directed graph, R the set of resources and N the set of architecture nodes (i.e., clusters). $s : V \longrightarrow \mathbb{R}^{+|R|}$ is a size function for the vertices, $q : A \longrightarrow \mathbb{R}^{+}$ is an affinity function for the weights of the edges and $C \in \mathbb{R}^{+|R|}$ a multi-dimensional array for the capacities of the nodes.
- Find an assignment of vertices to nodes, denoted $f : V \longrightarrow N$, which satisfies the capacity constraints

$$\sum_{v \in V : f(v) = n} s(v) \leq C_r, \forall n \in N, \forall r \in R,$$

and minimizes the objective function

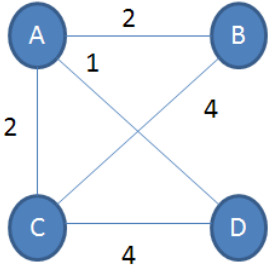
$$\sum_{a=(v,w) \in A : f(v) \neq f(w)} q(a).$$

Partitioning : an example



A graph example

- Mono-resource case
- Unitary weights for the vertices
- Unitary weights for the edges
- $C_{nr} = 5, \forall n \in N$



Partitioning (cost 13).

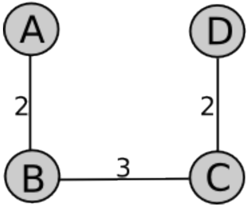
Relative affinity

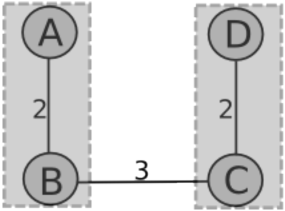
1. Relative affinity of $S \subset V$ to $S' \subset V$ ($S \cap S' = \emptyset$) :

$$\propto \alpha(S,S')\left(\frac{1}{\alpha(S,\bar{S})} + \frac{1}{\alpha(S',\bar{S}')}\right),$$

where $\alpha(S,S') = \sum_{a\in\delta(S,S')} q_a$.

2. Example of partitioning using relative affinity :



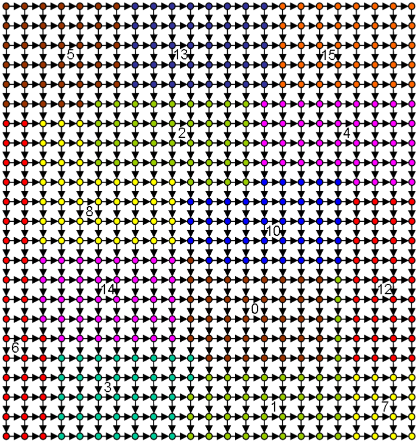


2-partition (cost 3).

A randomized greedy algorithm for the deterministic case

Algorithm 1 RG_PART

1. Initialization $W = V$
2. Assign the first $\min(|V|, |N|)$ vertices to the $|N|$ nodes and update W
3. Find an admissible assignment (v^*, n^*), if any, with max. relative affinity γ_1
4. Find an admissible fusion (n_1^*, n_2^*), if any, with max. relative affinity γ_2
5. If $\gamma_1 \geq \gamma_2$ then assign v^* to n^* and update the set W. Else merge n_1^* and n_2^*
6. If W is empty or there is neither any admissible assignment nor any admissible fusion, stop. Else, go to Step 3.



Partitioning of a grid 23×23.

Uncertainty sources and motivation

- Main sources of uncertainty for the partitioning problem : computing kernel execution times
- Causes :
  - Characteristics of the processor architecture (cache memories, memory access controllers)
  - Internal uncertainty due to execution times for computing kernels of intermediate granularity
    - Uncertainty due to data dependent control flows
- Characteristics :
  - Bounded support for the distributions of execution times
  - Multimodal distributions
  - Dependencies between the random variables (e.g. a target tracking pipeline)
- Difficult to :
  - Fully describe or estimate the parameters of such distributions
  - Make use of existing optimization techniques for dealing with such uncertainty

Related works : Optimization under uncertainty

Chance-constrained programs

$$\min_x f(x)$$
$$\mathbb{P}(G(x,\xi) \leq 0) \geq 1-\varepsilon.$$

- Issues
  - Combinatorial programs
  - Possible non-convexity of the feasible set
  - Complex probability distributions
- Resolution techniques
  - Approaches which guarantee to find optimal solutions
    - Convexities studies (e.g. Prékopa)
    - Relaxation methods for obtaining equivalent deterministic programs (e.g. Bertsimas et Sim, Ben-Tal et. Nemirovski)
    - Sampling approaches (e.g. Calafiore, Ahmed, etc.)
  - (Meta)Heuristics
    - Genetic algorithm and Monte-Carlo simulation (Loughlin)
    - Tabu search (Aringhieri, Tanner)
    - Beam search heuristic (Beraldi)

Robust binomial approach 1/2

- The weights of the tasks, depending of execution times are random variables (r.v.) with complicated multi-dimensional multi-modal joint distributions.

=>Non parametric sample approach using statistical hypothesis testing and an already existing algorithm for the deterministic case.

- No hypothesis made on the distribution of the random data (especially concerning the dependence between the r.v.)
- Easy adaptation of a heuristic already conceived for the deterministic version of the same problem
- Take advantage of available experimental data
- Elementary tools from statistical hypothesis testing theory

Robust binomial approach 2/2

- Let $\xi_1, \ldots, \xi_{NS}$ be a sample of size NS of i.i.d. realizations of the random vector ξ
- Approximation equivalent of a chance-constrained program :

Initial

$$\min_x f(x)$$
$$\mathbb{P}(G(x,\xi) \leq 0) \geq 1-\varepsilon.$$

Approximation

$$\min_x f(x)$$
$$\sum_{i=1}^{NS} \chi_i \geq k(NS, 1-\varepsilon, \alpha)$$
$$G(x, \tilde{\xi}_i) \leq (1-\chi_i)L; i = 1, \ldots, NS$$

- Binary variable χ_i for observation ξ_i :

$$\chi_i = \begin{cases} 1 & \text{if } G\left(x, \tilde{\xi}^i\right) \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

- $\sum_{i=1}^{NS} \chi_i \sim \mathscr{B}(NS, p_0)$ => determine $k(NS, 1-\varepsilon, \alpha)$ such that $p_0 = 1-\varepsilon$
- Parameter $\alpha \in (0,1)$, confidence level, is the type I error of the statistical hypothesis test :

$$\begin{cases} \mathrm{H}_0 & : & \mathbb{P}(G(x,\xi) \leq 0) < 1-\varepsilon \\ \mathrm{H}_1 & : & \mathbb{P}(G(x,\xi) \leq 0) \geq 1-\varepsilon \end{cases}$$

Randomized greedy algorithm for the stochastic case (1/2)

- If the weights of the vertices S_{vr} are r.v., then the capacity constraints

$$\sum_{v \in V : f(v) = n} s(v) \leq C_r, \forall n \in N, \forall r \in R$$

become

$$\mathbb{P}\left(\bigwedge_{n \in N} \bigwedge_{r \in R} \sum_{v \in V : f(v) = n} S_{vr} \leq C_r \right) \geq 1 - \varepsilon.$$

- Adaptation of the randomized greedy algorithm for the stochastic case
  - Compare the number of constraint violations to $k(NS, 1-\varepsilon, \alpha)$
  - Modification of the notions of admissible assignment and of admissible fusion
  - Respect the prescribed probability level ε with a given confidence level α
- Complexity : a linear increase with a factor of NS

Randomized greedy algorithm for the stochastic case (2/2)

Algorithm 2 RG_PART_STOCH

Input: W, N, R, ε, α, NS, $\tilde{S}_{vr}^{(k)}$ for each $\{v \in V,\ r \in R,\ k = 1...NS\}$

1: Initialization $W = V$
2: Assign the first $\min(|V|, |N|)$ vertices in lexicographic order to the $|N|$ nodes and update the set W
3: Find an **admissible stochastic assignment** (v^*, n^*) $(v^* \in W,\ n^* \in N)$, if any, with max. relative affinity γ_1
4: Find **admissible stochastic fusion** (n_1^*, n_2^*) $(n_1^* \in N,\ n_2^* \in N)$, if any, with max. relative affinity γ_2
5: If $\gamma_1 \geq \gamma_2$ then assign v^* to n^* and update the set W. Else merge n_1^* and n_2^*.
6: If W is empty or there is neither any admissible assignment nor any admissible fusion, stop. Else, go to Step 3.

Benchmark

- Uncertain Parameters Generation
  - Simulation of a joint bimodal distribution (modes uniform in their intervals and selected in an equally likely manner)
    - 1st mode : hypercube $[0,8\,;0,9]^{|V|}$
    - 2nd mode : hypercube $[1,1\,;1,2]^{|V|}$
  - Sample size : 100 or 1000
- Data sets (modified for stochastic case)
  - Examples of grids, representative in size for our application
  - Johnson et al. bisection instances
  - Mono-dimensional resource
- Deterministic case
  - Unitary weights for vertices and edges
  - Average differential approximation ratios of 5.22% compared to best known solutions
- Evaluation of stochastic algorithm
  - Overall : cost, robustness, computation time
  - Test 1 : number of nodes (same node capacity)
  - Test 2 : capacity of nodes (same number of nodes)

Results - stochastic version (1/3)

Table : $NS = 100,\ \varepsilon = 0.05,\ \alpha = 0.05$

	1st test			2nd test		
Name	#nodes	sol	time	C	sol	time
Grid 4×4	6 (4)	14 (8)	≈ 0	4.71 (4)	12	≈ 0
Grid 10×10	6 (5)	38 (28)	0.02 s	23.3 (20)	29	0.01 s
Grid 23×23	16 (16)	182 (150)	1.12 s	44.1 (40)	173	0.99 s

Table : $NS = 1000,\ \varepsilon = 0.01,\ \alpha = 0.01$

	1st test			2nd test		
Name	#nodes	sol	time	C	sol	time
Grid 4×4	6	14	≈ 0	4.74	10	≈ 0
Grid 10×10	6	37	0.15 s	23.36	37	0.13 s
Grid 23×23	16	182	10.75 s	44.183	193	9.67 s

Results 2nd data set - stochastic version (2/3)

- Johnson instances
  - 25 graphs with the number of vertices varying between 124 and 1000
  - Tests with samples of 100 ($\varepsilon = 0.05$, $\alpha = 0.05$) and 1000 ($\varepsilon, \alpha \in 0.01, 0.05$)
- Results
  - 14 and respectively 15 robust solutions with a gap in the cost quality of less than 5% from the deterministic solutions for $\varepsilon = 0.05$, $\alpha = 0.05$
  - 14 robust solutions with a relative 5% gap in the cost quality to the deterministic solutions for $\varepsilon = 0.01$, $\alpha = 0.01$
  - Execution times : an average of 48.04 sec. for a sample size of 1000 against 25.93 sec. for a sample size of 100

Overall results - stochastic version (3/3)

- Similar results when increasing the sample size
- 1st test
  - Ratio of 1.5 between the number of nodes needed for finding a stochastic feasible solution and the number of nodes in the deterministic case for both data sets
- 2nd test
  - Equally large increase in the capacity of the nodes in the order of 1.1
- Acceptable execution time
- Solutions of good quality, statistically significantly guaranteed (α) with a target probability level (ε)
- Solutions found by the algorithm for the deterministic case not robust in $\approx 50\%$ of cases

Conclusion et Perspectives

- Conclusion
  - Problem of chance-constrained partitioning networks of communicating processes arising in dataflow compilation
  - A qualitative analysis of the sources of uncertainty lead to the choice of a non parametric approach
  - Heuristic algorithm combining a sample statistical approach with an existing (software engineering consideration) greedy method for the deterministic version
  - Statistically significantly **robust** solutions of an acceptable quality within an admissible average running time
- Perspectives
  - Design of a parallelized implementation of the method
  - Apply the robust binomial approach to other optimization problems related to the compilation for manycore systems (e.g. placement/routing)

Thank you!
Questions?