Goodness of Time-Processor Optimal PRAM Simulations

Ville Leppänen

Department of Computer Science, University of Turku, Finland

Abstract. We address the question 'how to measure goodness of time-processor optimal PRAM simulations'. Instead of measuring only the asymptotic complexity of simulation time, we attempt to take into account all aspects of simulations exactly. We present a goodness function framework and propose a generic function for measuring the goodness.

1 Introduction

A simulation of an N-processor PRAM on a P-processor distributed memory machine (DMM) is *time-processor optimal*, if simulation of a PRAM step succeeds in time $O(N/P)$ (with high probability). The simulation time is lower bounded by the diameter ϕ of the routing machinery and the expected memory congestion γ. If the DMM is symmetric and memory requests can be satisfied by only one memory module (one hash function), expected length δ of memory request route is $\delta = \Theta(\phi)$. If the total routing capacity is ψP (ψ packets per physical processor per step), two necessary conditions for time-processor optimality are that the load $\ell = N/P$ (parallel slackness factor) is $\ell = \Omega(max\{\phi, \gamma\})$ and $\psi = \Omega(\phi)$. Many such solutions can be derived [3, 4, 5, 6, 9].

Often when simulation results are reported, the asymptotic complexity of simulation time is highlighted while other aspects are almost ignored. "Better" results can be obtained by assuming stronger graph theoretical properties (larger degree Δ, smaller ϕ) and/or stronger (shared resource) contention resolution protocols. Comparison of PRAM simulations should account for implementation of routing machinery properties, since a PRAM simulation is basically a routing problem! How to account for routing machinery implementation? Using the VLSI cost model to implement communication graphs has been studied extensively. The basic VLSI components are transistors and wires, and the success in modeling the cost leans on the fact that the size of basic components is approximately the same. Extension to processor&memory modules and connections between them is possible, but the size issue can be severely off balance. Using optics instead of VLSI to implement communication has been suggested in [7, 10] – more or less in the form of static multichannel mesh of optical buses.

A $SMcMOB(s, d, \rho)$ consists of a d-dimensional mesh of optical buses – the side length along each axis is s nodes. Each node is connected to a bus along each axis with ρ receivers and transmitters. Each transmitter (receiver) connected to a bus is assumed to have a fixed unique channel. (In [8], a hardware configuration capable of supporting 250 channels per bus is discussed. Each of them operates

at rate $\approx$ 1 Gbit/s. The configuration is claimed to be able to support up to 5000 channels.) We assume that a PRAM simulation $\mathcal{S}$ is implemented on a logical architectural model (of a routing machinery) which in turn is implemented by embedding it into a $SMcMOB$. Our goodness function parameters come from simulation, architectural model and embedding of the communication graph.

An *embedding* $\mathcal{E} = (G, H)$ of graph G into $H = SMcMOB(s, d, \rho)$ assigns nodes of G on nodes of H; maps a channel for each transmitter and receiver properly; and sets a path to each directed edge of G. A stepwise *emulation* $\mathcal{F}$ assigns a collision-free schedule for the movement of packets according to the paths. Let φ_t denote the length of schedule. Besides φ_t, we are interested of stretch φ_s, which is the largest stretch of a channel in H. If a channel connects nodes $\langle x_1, \ldots, x_i, \ldots, x_d \rangle$ and $\langle x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_d \rangle$, the stretch of channel is $|x_i - y_i|$. Above kind of embedding and emulation has been studied in [7, 10].

2 Goodness functions

Let $\mathcal{I}$ be an implementation of an EREW simulation $\mathcal{S}$ running on a DMM $\mathcal{M}$ so that $\mathcal{I}$ is based on an emulation $\mathcal{F}$ on an embedding $\mathcal{E} = (\mathcal{M}, SMcMOB(s, d, \rho))$.

Informally our *goodness functions* map a certain set of PRAM implementation related parameters to scalar values. The best value a goodness function can give for an implementation is 1. Besides φ_t and φ_s, we see the following parameters of $\mathcal{M}$ and $\mathcal{S}$ relevant: the number of processors and memory modules P; the number of routing machinery nodes Q; the routing machinery communication capacity Φ (expressed in packets); the logical diameter ϕ; the maximum degree Δ of nodes and processor&memory modules; the size of queues q; the size of packets z; and the load-cost function of simulation C. The function C measures simulation cost per simulated PRAM processor on load ℓ. If $\mathcal{S}$ uses at most $T(N, P)$ steps to simulate an N-processor PRAM step on a P-processor DMM $\mathcal{M}$ (with high probability), the *load-cost function* of $\mathcal{S}$ is $C(\ell) = T(N, P)/\ell$.

2.1 General framework

In our model the cost of a PRAM implementation $\mathcal{I}$ comes from (i) *extraneous hardware* needed to implement communication (useless from the calculation point of view); (ii) *unit step* concept; and (iii) PRAM *simulation algorithm*. The total cost $\mathcal{G}(\mathcal{I})$ we form by multiplying the cost of individual factors.

We model the total cost of hardware by $\mathcal{C}_{iron}(\mathcal{I}) = \mathcal{C}_{P+M} + \mathcal{C}_{Q,\Delta} + \mathcal{C}_{\Phi,q,z}$, where $\mathcal{C}_{P+M}$ is the cost of processor&memory modules; $\mathcal{C}_{Q,\Delta}$ is the cost of routing machinery nodes (their internal logic); and $\mathcal{C}_{\Phi,q,z}$ is the cost of communication capacity (hardware cost of queues + network interfaces). The cost of extraneous hardware is obtained from $\mathcal{C}_{iron}(\mathcal{I})$ by dividing it by basic costs: $\mathcal{C}_{P+M}$.

Several factors put stress on the concept of unit time. We assume that in a time unit a packet can be moved to any neighboring node (in $\mathcal{M}$) and over each directed connection; and the routing machinery can accomplish one elementary

routing round. Moreover, the amortized packet processing time (fetch/store, simple arithmetic operation, fetch next operation, calculate addresses, ...) should not exceed the unit time. Therefore, $\mathcal{C}_{step}(\mathcal{I}) = max\{\mathcal{C}_{s,z,t}, \mathcal{C}_{M,P}, \mathcal{C}_{\Delta}\}$, where $\mathcal{C}_{s,z,t}$ represents the cost due to stretch of embedding, size of packets, and emulation time; $\mathcal{C}_{M,P}$ represents the unit time cost of processor&memory modules; and $\mathcal{C}_{\Delta}$ models the functional complexity of underlying routing machinery nodes.

The cost of PRAM simulation algorithm we model by $\mathcal{C}_{sim}(\mathcal{I}) = \mathcal{C}_C \times \mathcal{C}_{\phi}$, where $\mathcal{C}_C$ charges for the properties of the load-cost function in general and $\mathcal{C}_{\phi}$ represents cost for the slackness needed to achieve certain simulation cost level.

In our opinion, $\mathcal{G}(\mathcal{I})$ allows us model problems related to (a) the concept of unit time, (b) the implementation of a topology, (c) the properties of load-cost function C, and (d) the load requirement. The shape of C is not everything that matters and the cost of extraneous hardware can be put into right perspective.

2.2 A generic instance of the framework

We reduce the number of parameters to four by introducing a generic goodness function $g_{\alpha,\beta,\gamma,\omega}$. We model processor and memory module hardware costs by $\mathcal{C}_{P+M} = P \times C'_{P+M}$, where C'_{P+M} is the hardware cost of a processor& memory module. We assume that routing machinery nodes are homogeneous and model $\mathcal{C}_{Q,\Delta} = Q(H) \times C'_Q \times C'_{\Delta}/4$ and $\mathcal{C}_{\phi,q,z} = \rho \times d \times Q(H) \times C'_{q,z}$, where C'_Q is hardware cost of the routing logic of a simple router node. We take a binary tree node as a reference node. The C'_{Δ} expresses the maximum number of "paths" form incoming edges to outgoing edges via a node (4 for a binary tree node). We expect the cost of router node logic to grow linearly with C'_{Δ}. The $C'_{q,z}$ is hardware cost of a buffer of size q and width z plus the cost of a network interface ($\approx$ a transmitter and a receiver). The ratio of extraneous hardware is now

$$\frac{\mathcal{C}_{Q,\Delta} + \mathcal{C}_{\Phi,q,z}}{\mathcal{C}_{P+M}} = \frac{Q(H) \times C'_Q \times C'_{\Delta}/4 + \rho d \times Q(H) \times C'_{q,z}}{P \times C'_{P+M}} \leq (\frac{C'_{\Delta}}{4} + \rho d)\frac{Q(H)}{P} \times \alpha,$$

where $\alpha = max\{C'_Q, C'_{q,z}\}/C'_{P+M}$ represents the hardware ratio of simple routing machinery element and processor&memory module.

What can we say about α? Experiments show that $q = 4, \ldots, 8$ is sufficient, and in simulation algorithms z is at most $\approx$ 250b. Since typically tens or hundreds of megabytes of memory is attached to a processor and processors contain several million VLSI components, we claim that the ratio C'_Q/C'_{P+M} can well be $1/10^5$. The prevailing idea is that the network interface is based on optics – we find it difficult to estimate the complexity of such an interface. In our opinion, α can be $1/10^4$, or more, but the "current" value of α is obviously much less.

How to model $\mathcal{C}_{step}$? Let $\mathcal{C}_{M,P} = 1$. The clockrate of processors is currently 0.1GHz – 0.5GHz. Techniques to push amortized memory access time close to cycle time exist. We model $\mathcal{C}_{\Delta}$ in terms of C'_{Δ}/ω. For Tera [2], $C'_{\Delta} \approx 20$. Although routing through a node requires 3 cycles in Tera, we take 20 to be a good candidate for ω. Tera can move a packet over each connection on every clock

tick. We conclude that $C_{s,z,t}$ depends mainly on φ_s and φ_t, and propose formula $C_{s,z,t} = \varphi_t \times \varphi_s/\beta \times \lceil \sqrt[d-1]{\rho/2} \rceil$, where $\lceil \sqrt[d-1]{\rho/2} \rceil$ represents the unit length of stretch (side length s_n of a router node) and β represents the amount of nodes a signal can pass by in a time unit. If $s_n \approx$ 1cm, the current clockrates suggest that β can be $\approx 10^2$. The value of β has a great impact on the shape of ideal routing machinery, since the length of free "jumps" depends on it.

Next we model C_C and C_ϕ. According to our experience, on a DAG the load-cost function $C(\ell)$ of a greedy routing + synchronization wave based simulation is of the form $C(\ell) \approx b_\ell \times max\{\ell, \phi_r\} + O(1)/\ell$, where ϕ_r is the length of longest "source $\mapsto$ target $\mapsto$ source" path and b_ℓ is almost a constant. We set $C_C = b_{2\phi_r}$, since b_ℓ often approaches 1 freely and is close to its best when $2\phi_r \leq \ell \leq 5\phi_r$.

What is the cost C_ϕ of slackness (e.g., $\approx 2\phi_r P$) required to achieve a good simulation cost considering machine design and usage? Tera supports 128 threads whereas SB-PRAM [1] supports 32 × 32 threads. Representation of a thread requires $\approx$ 1–3Kb – obviously hardware support for a large number of threads is possible. More essential is that the operating system (OS) is able to provide enough threads for each physical processor. The OS can do this via concurrent applications, but luckily several common problems (e.g., matrix multiplication and sorting) have work-optimal NC-class EREW solutions. This suggests that applications can provide a large number of threads. In our opinion, the cost of slackness ℓ_0 is not ℓ_0 but much less. Should a sublinear function be used and/or should some threshold value be applied to ℓ_0? We model $C_\phi = \gamma(2\phi_r)$. Choosing a fair γ is difficult, but we find tempting to try, e.g., $\log_2 x$ and $max\{1, x/10\}$.

References

1. F. Abolhassan, R. Drefenstedt, J. Keller, W. Paul, and D. Scheerer. On the Physical Design of PRAMs. *The Computer Journal*, 36(8), 756 – 762, 1993.
2. R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The Tera Computer System. *Comp. Arch. News*, 18(3):1 – 6, 1990.
3. A. Czumaj, F. Meyer auf der Heide, and V. Stemann. Shared Memory Simulations with Triple-Logarithmic Delay. In *Proceedings of ESA '95*, 46 – 59, 1995.
4. L. Goldberg, Y. Matias, and S. Rao. An Optical Simulation of Shared Memory. In *SPAA '94, Symposium on Parallel Algorithms and Architectures*, 257 – 267, 1994.
5. F. Leighton, B. Maggs, A. Ranade, and S. Rao. Randomized Routing and Sorting on Fixed-Connection Networks. *J. of Algorithms*, 17(1):157 – 205, 1994.
6. V. Leppänen and M. Penttonen. Work-Optimal Simulation of PRAM Models on Meshes. *Nordic Journal on Computing*, 2(1):51 – 69, 1995.
7. W.F. McColl. General Purpose Parallel Computing. In *Proceedings of 1991 ALCOM Spring School on Parallel Computation*, 337–391, 1993.
8. A. Nowatzyk and P. Prucnal. Are Crossbars Really Dead? The Case for Optical Multiprocessor Interconnect Systems. *Comp. Arch. News*, 23(2):106 – 115, 1995.
9. A.G. Ranade. How to Emulate Shared Memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.
10. S.B. Rao. Properties of an Interconnection Architecture based on Wavelength Division Multiplexing. Report TR-92-009-3-0054-2, NEC Research Institute, 1992.