Scalability of Architectures and Algorithms under Fundamental Constraints

Gianfranco Bilardi

University of Padova

Imperial College, September 9, 2024

NANDA'24 - Novel Architectures and Novel Design Automation

Processing Bits vs. Transferring Bits

Theorem

For most computations, asymptotically,

- As technology approaches the limits imposed by physical constraints on lower bounds to *device size* and upper bounds to *message speed*,
- and as systems of larger scale are built:

1. the fraction of cost due to processing information approaches **0**;
2. the fraction of cost due to transporting information approaches **1**.

Paradigm Shift

Processing-centric → Communication-centric

- **Algorithm design and implementation** switch minimization focus from the number of operations to the communication requirements.
- **Machine architecture and organization** switch optimization focus from ALU operations to data transport.

Avenues to reduce impact of communication on computation time:

- Exploit locality to decrease the distance travelled by messages.
- Exploit **concurrency** to increase the number of messages travelling at the same time.

Three Major Barriers

In past and current developments *technological* limitations have amplified the fundamental constraints, creating three major performance barriers:

1. The Memory Wall
2. The Chip Boundary Wall

The Memory Wall

The Random Access Machine

The RAM is an abstract model of computation, the basis for most sequential algorithmics.
It is an idealized version of the von Neumann architecture, the basis for much architectural developments.

Key Assumptions

- One operation per step, including operand load and store.
- Constant time per step.

Question

- Is constant time per step feasible for a physical machine?

The (von Neumann?) Bottleneck

A Current Engineering Problem

- Memory latencies are large, measured in processor cycles.
  Main memory latency = $O(100)$ cycles;
  Disk latency = $O(10^6)$ cycles.
  $\Rightarrow$ The CPI (Cycles Per Instruction) metric increases.

A Fundamental Problem

- From the principles of Maximum Information Speed and Maximum Information Density it follows that

  In any physical realization of the RAM with S bits of memory, in d dimens., one step must take $\Omega(S^{1/d})$ time, on aver., to complete.

Question

How small a CPI can a physical machine achieve, as a function of S, for arbitrarily large program/data sizes?

Framework and Approach (1)

Design Rules

- Proposed machine organizations have a layout such that number of gates and geometric distance traversed by a signal in one cycle are costant, *i.e.,* independent of machine size.

Memory Organization

Pipelinable: to overlap latencies, when possible to exploit access concurrency.

Hierarchical: to reduce latency, when possible to exploit temporal locality.

Processor Organization

Speculative and Parallel: to exploit operation and access concurrency (static and dynamic).

Memory Bypass: to avoid memory latency for some of the accesses.

Framework and Approach (2)

Program Structure

Results on achievable performance are obtained under suitable assumption on the structure of the RAM programs, in terms of dependences:

Direct-Flow Programs: only functional dependences.

Straight-Line Programs: functional and address dependences.

General Programs: functional, address, and control dependences.

Further program properties are captured by the *Latency Graph*.

Overview of Results (1)

Direct-Flow Programs: $CPI = O(1)$

- Pipelinable memory
- Speculative prefetch processor with memory bypass

Straight-Line Programs: $CPI = O(AD{-}DEPTH)$

- Pipelinable memory
- Speculative prefetch & execute

General Programs $CPI = O(H_{LG})$

- Pipelining instructions over predicted paths (with code motion)
- Hierarchical pipelinable memory
- Speculative prefetch & execute & branch

Overview of Results (2)

Some Algorithmic Bounds

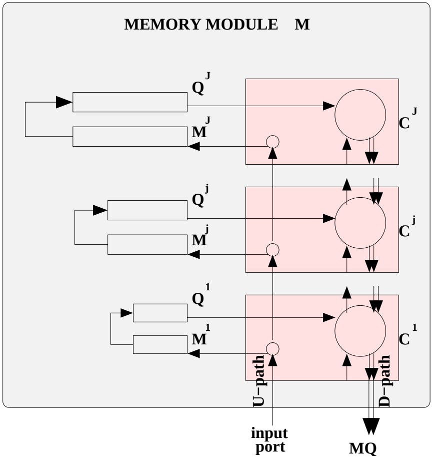
- matrix $+$ & $\times$; bitonic merging & sorting; FFT; digital filtering; finite-difference solvers for some PDEs: $CPI = O(1)$ (*just memory pipelinability*)
- quicksort, Strassen matrix $\times$: $CPI = O(1)$ (*memory pipelinability + hierarchy*)
- Merging (standard seq. alg.): $CPI = O(\log\log N) \quad (d = 2, 3)$

Lower Bound Methodology for a Class of Machines

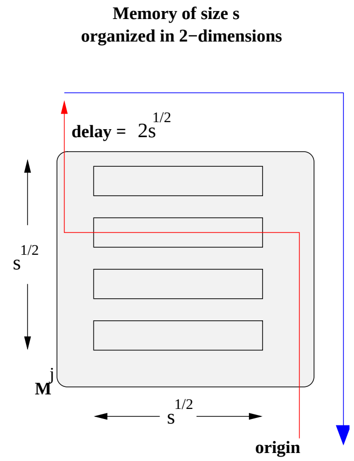
- Concept of Literal Execution
- There are direct-flow programs with $CPI = \Omega(\log\log N)$

Memory Organization

There are **scalable** d-dimensional memories with *optimal access time* $a(x) = O(x^{1/d})$ *pipelineable at a fixed rate*.

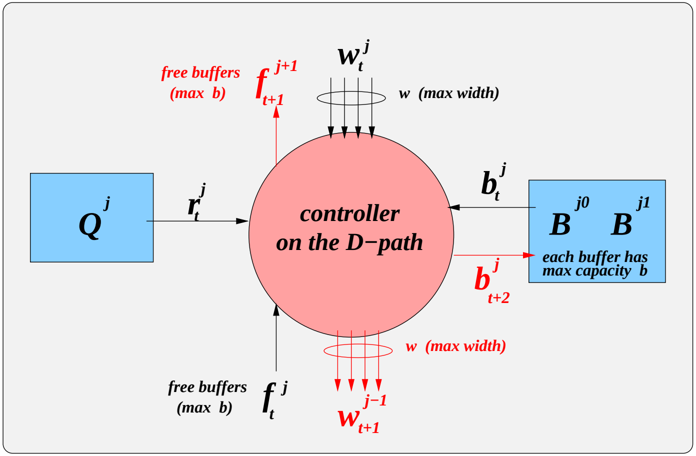


Memory Organization



Non-Hierarchical Pipelined Module

Memory Organization



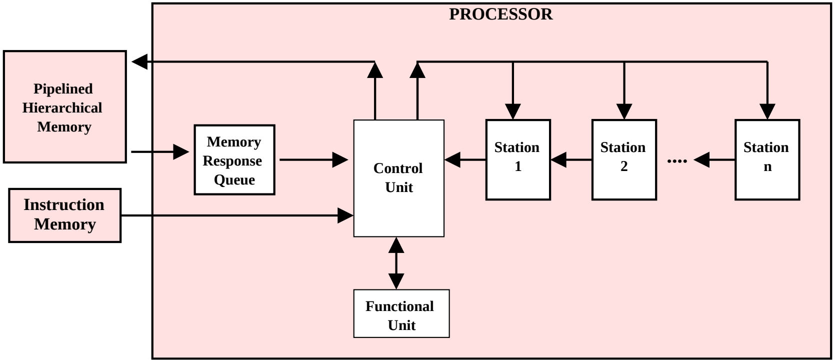
Overview of Controller in the D-Path

Memory Organization

Rules of Controller in the D-Path

$$\begin{aligned}
r^j_{max,t} &= f^j_t + [w - (w^j_t + b^j_t)], && (1)\\
r^j_t &= \min(q^j_t, r^j_{max,t}), && (2)\\
w^{j-1}_{t+1} &= \min(r^j_t + w^j_t + b^j_t, w - b + f^j_t), && (3)\\
b^j_{t+2} &= r^j_t + w^j_t + b^j_t - w^{j-1}_{t+1}, && (4)\\
f^{j+1}_{t+1} &= b - b^j_{t+2}. && (5)
\end{aligned}$$

Speculative Prefetcher Processor



Speculative Prefetcher Processor

Execution Strategy

- Prefetches $n = O(max - latency)$ instructions along (statically) predicted paths. Code motion based on a path separator of the CFG guarantees pipelinability.
- Prefetches data speculatively (*i.e.,* not updated by the current instruction window.
- Rounds ($\leq n$): executes on speculative data and control and checks for cosistency of results; if inconsistency, refetches and reexecutes.

The Chip Boundary Wall

The Chip Boundary Wall

Problem

Chip I/O bandwidth a fraction $10^{-2} \div 10^{-3}$ of the internal bisection bandwidth.

Question

Given a target computation, what fraction of the chip area can be devoted to “sustainable” functional units?

Remark. For many computations, the above fraction vanishes asymptotically. But, what about current and forthcoming technologies?

On-Chip Processing: Tradeoff Formulation

Relevant Quantities

- n: size of subproblem mapped onto chip;
- $w(n)$: number of operations per subproblem;
- m: on-chip memory (bit);
- $I(n, m)$: information exchange across chip boundary, with memory and other processing chips (bit);
- b: chip I/O bandwidth (bit/cycle);
- F: operations/cycle.

On-Chip Processing: Tradeoff Formulation

Balance Equation

$$w(n)/F = I(n,m)/b. \tag{6}$$

Area Budget Equation

$$A_F\ F + A_m\ m = \eta A. \tag{7}$$

The solution to the above system of equations provides the optimal area split between memory and functional units.

Analysis of QCD (Dirac Operator)

Computation Requirements

- $n = k^3 N_4$: sublattice points
- $w(n) = \chi n$, $(\chi = 2324)$;
- $m = L_w \sigma n$ $(L_w = 64$, $\sigma = 120)$;
- $I(n, m) = \iota n/k$ $(\iota = 288)$.

Technology Assumptions

- $A = \ell^2 \lambda^2$, (λ feature size);
- $A_F = \alpha_F \lambda^2$, $(\alpha_F = 10^8)$;
- $A_m = \alpha_m \lambda^2$, $(\alpha_F = 50$, embedded DRAM);
- $b = \beta 4 \ell$ $(\beta = 1/3000)$.

Key Equation

$$\alpha_F F + \alpha_m m = \gamma \ell k + \delta N_4 k^3 = \ell^2, \tag{8}$$

The Network Wall

Communication Cost

- Communication heavily affects the efficiency of parallel algorithms
- Communication costs depend on interconnection technology, bandwidth, latency, ...

Question

- Can we design efficient parallel algorithms oblivious to machine structure and parameters?

Cache-Oblivious Framework

Specification model:

CPU+RAM

Evaluation model:

CPU + (M,B)-CACHE + RAM

Execution model:

CPU + (M_1,B_1)-CACHE + (M_2,B_2)-CACHE + $\cdots$ + RAM

Cache-Oblivious Algorithms

- Algorithm does not use M or B
- Optimality in evaluation model $\forall(M,B) \Rightarrow$ optimality in execution model $\forall(M_1, B_1, M_2, B_2, \ldots)$

Network-Oblivious Framework

Specification model:

Linearly ordered set of n *virtual* processors (n function of input size only)

Evaluation model:

Linearly ordered set of p *physical* processors and communication block size B

Execution model:

$$\text{D-BSP}(p; g_1, g_2, \dots; B_1, B_2, \dots)$$

Hierarchical network of p *physical* processors where clusters of size $p/2^i$ have *inverse bandwidth* g_i and *block size* B_i

Network-Oblivious Algorithms

Fundamental Theorem

Optimality for the evaluation model

$\Rightarrow$

Optimality for the execution model

$\Rightarrow$

Optimality for several common topologies, including d-dimensional arrays

Case studies

- Matrix Multiplication and FFT do admit optimal network-oblivious algorithms
- Matrix Transposition does not admit an optimal network-oblivious algorithm

Future Directions

Parallel Pipelined Hierarchical Systems

Can we establish upper and lower limits to the achievable CPI?

Chip Boundary and Information Exchange

Can we formulate a systematic theory of accelerators?

Network Obliviousness

Can we characterize the properties of computations for which there are oblivious algorithms?

Acknowledgments

Memory Wall

Joint work with K. Ekanadham, P. Pattnaik (IBM Research)

Chip Boundary Wall

Joint work with A. Pietracaprina, G. Pucci (U. Padova) and F. Schifano, R. Tripiccione (U. Ferrara)

Network Wall

Joint work with A. Pietracaprina, G. Pucci, F. Silvestri (U. Padova)