Initial Results on Heterogeneity for Linear Algebra

Grey Ballard, James Demmel, **Andrew Gearhart**

UC Berkeley

Parlab Retreat
June 1, 2011

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung.

Communication-Avoiding (CA) work in Parlab

- Faculty: James Demmel, Armando Fox and Kathy Yelick
- President Obama cited communication avoiding algorithms in the FY 2012 Department of Energy Budget Request to Congress (regarding CA-GMRES work from Mark Hoemmen)

Ongoing Projects (with involved students)

- **Heterogeneous CA algorithms**: Grey Ballard and Andrew Gearhart
- **CA Successive Band Reduction**: Grey Ballard and Nick Knight
- **Hypergraph Partitioning**: Erin Carson and Nick Knight
- **CA Krylov Subspace Methods**: Erin Carson and Nick Knight
- **CA Solvers for Band Matrices**: Razvan Carbunescu
- **Lower bounds for Strassen**: Grey Ballard, Olga Holtz and Oded Schwartz
  - Best Paper Prize in SPAA'11
- **CA Gang-scheduling for multicore**: Juan Colmenares
  - Talk 2:00pm tomorrow!!
- **SEJITS Specializers**: Shoaib Kamil
  - Talk 9:00am tomorrow!!
- **2.5D Algorithms**: Edgar Solomonik
  - Distingished Paper Prize in EuroPar'11
  - Talk 9:00am on Friday!!
- **Topology-aware Collectives**: Edgar Solomonik
- **CA QR Algorithms**: Michael Anderson

Talk Summary

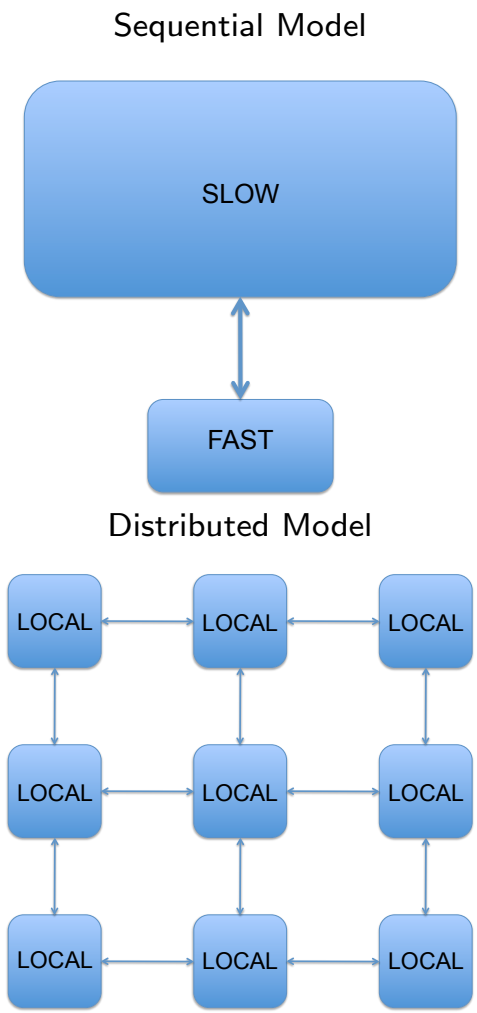
- **Communication-avoiding** algorithms move as little data as possible, since these are the slowest and energy-hungriest operations any computer performs
- **Model for Heterogeneous Processing** is presented and justified
- We extend previous work on **communication lower bounds** to our heterogeneous model
- **Communication-optimal heterogeneous algorithms** for matrix-matrix and matrix-vector multiplication are presented

Previous Models of Computation

- Machine models assume "fast" and "slow" types of memory access
- We wish to asymptotically minimize "slow" traffic

Previous Models of Computation

- Machine models assume "fast" and "slow" types of memory access
- We wish to asymptotically minimize "slow" traffic
- **Sequential**: Single processor is separated from memory via a small, fast cache ("fast" = cache, "slow" = DRAM)
- **Distributed**: A group of processors is connected on a network ("fast" =local access, "slow" = remote access)



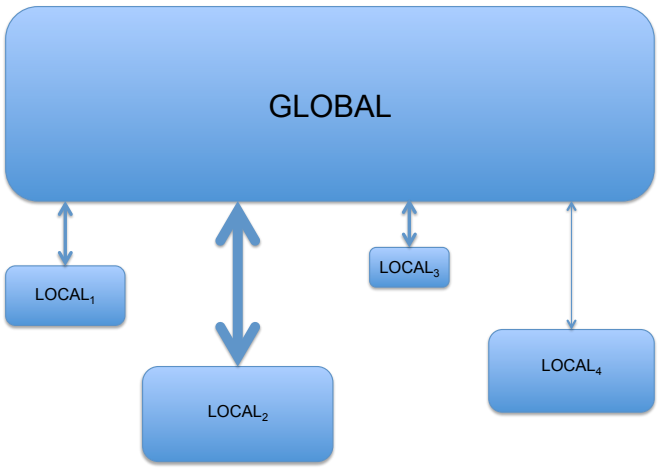
Previous Models of Computation

- Distributed model assumes homogeneous processors and network links
- Our heterogeneous model assumes a global shared memory with processing elements defined by a set of empirical parameters
- Goal of the model is to capture the design space of likely best algorithms...while leaving specific parameter selection to autotuners

Heterogeneous Model: Considerations

- For this work, we assume each processor to have an independent link to a shared global memory and that processing rate is constant.

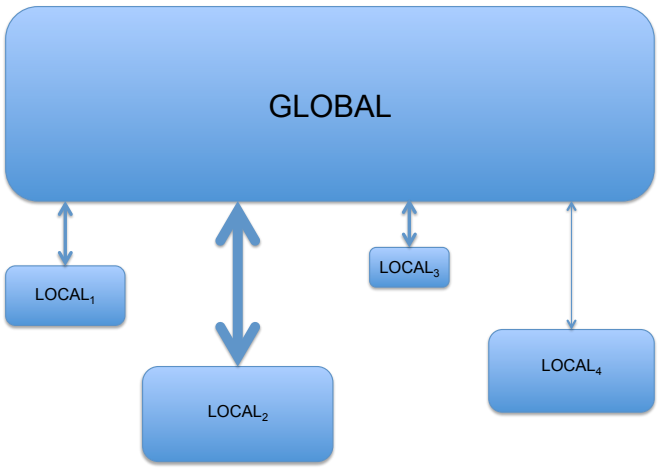
Heterogeneous Model



Heterogeneous Model: Considerations

- For this work, we assume each processor to have an independent link to a shared global memory and that processing rate is constant.

Heterogeneous Model



- A heterogeneous processing element i is defined by:
  - M_i: fast memory size (words)
  - γ_i: processing rate (sec/flop)
  - α_i: link latency between fast and slow memory (sec/msg)
  - β_i: link inverse bandwidth (sec/word)
- Processing elements can be CPU cores, GPUs, FPGAs, etc.

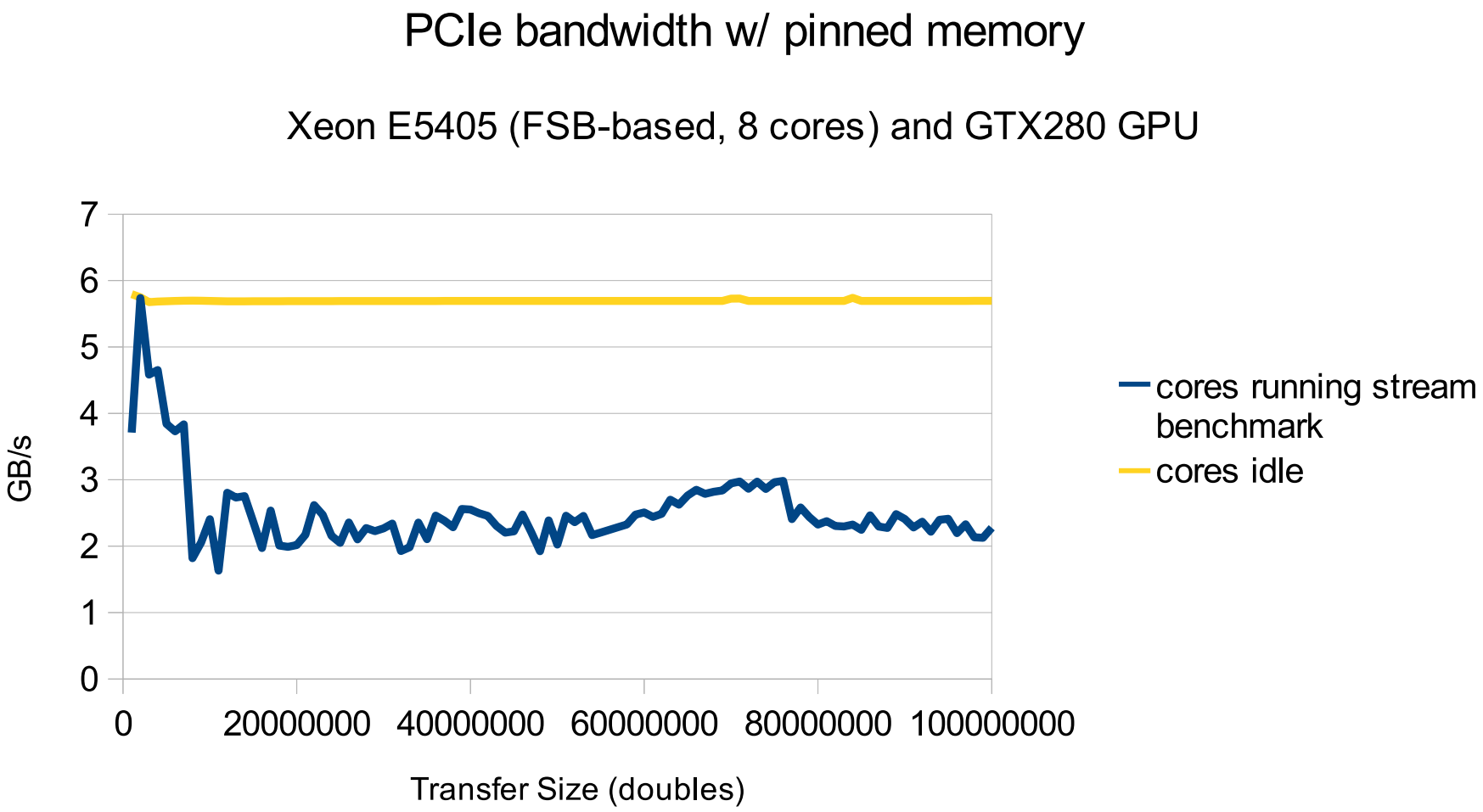
- But wait, Andrew. Are these assumptions accurate???

- But wait, Andrew. Are these assumptions accurate???
- Uh...maybe.

- But wait, Andrew. Are these assumptions accurate???
- Uh...maybe. Bandwidth and latency benchmarking show a more complex story...

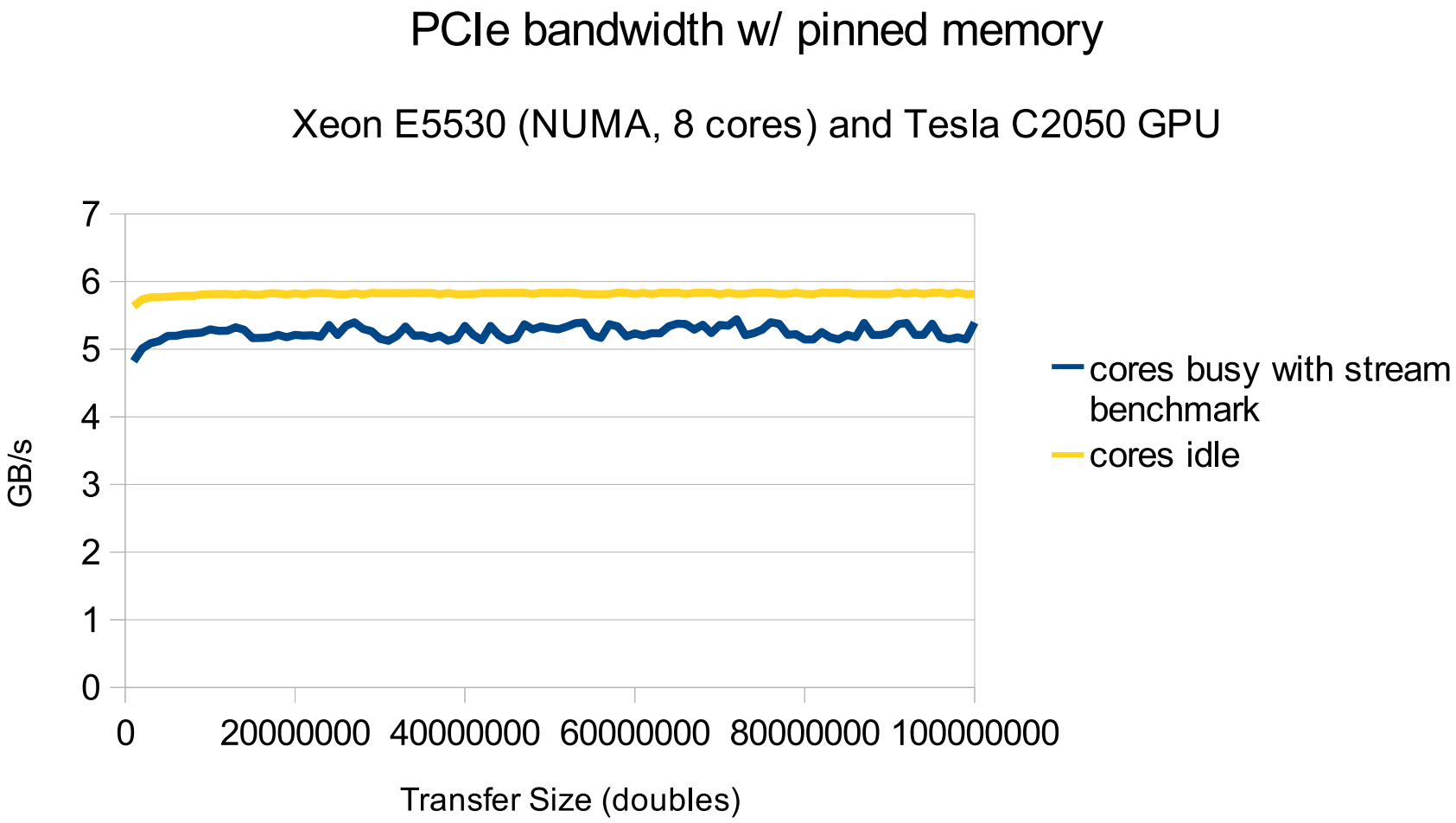
Heterogeneous Model: Considerations

- FSB-based machine:

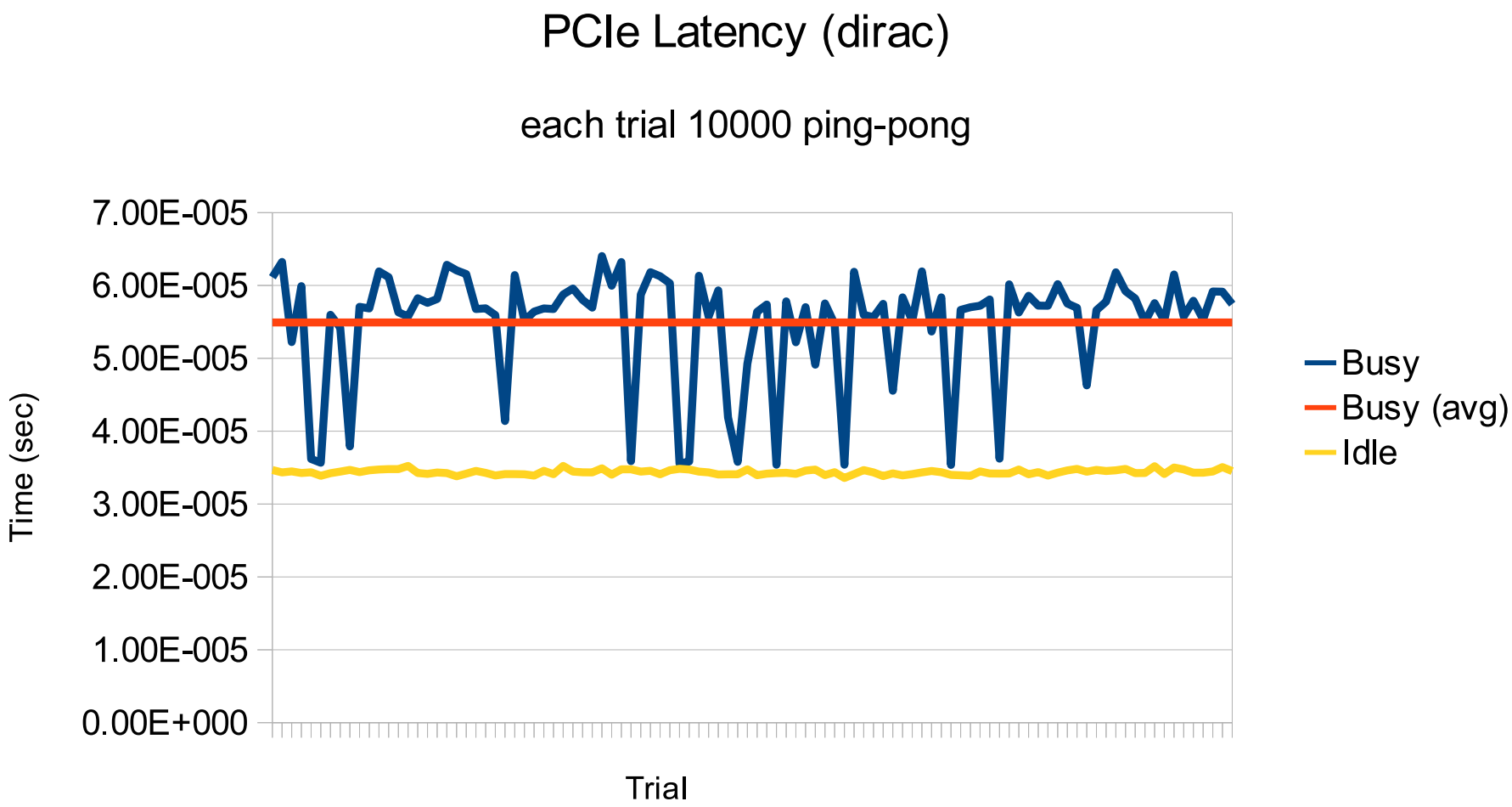


Heterogeneous Model: Considerations

- NUMA machine:



Heterogeneous Model: Considerations



Heterogeneous Lower Bounds : Execution Model

- Message transfer times are modeled as $T_{\mathbf{msg}} = \alpha + \beta W$
- We lower bound the parallel program's total runtime T by considering the last processor to finish execution:

$$T(\{F_i\}) \geq \max_{1 \leq i \leq P} \{\gamma_i F_i + \beta_i W_i + \alpha_i L_i\}$$

where for processor i:

F_i	number of flops
W_i	words transferred
L_i	messages sent
γ_i	processing rate
β_i	inverse bandwidth
α_i	message latency

Heterogeneous Lower Bounds : General Bound

- Previous bounds for serial model (Ballard et al. 2009):

$$W_i \geq \max\left\{I_i + O_i, \frac{F_i}{8\sqrt{M_i}}\right\}, L_i \geq \max\left\{\frac{I_i + O_i}{M_i}, \frac{F_i}{8M_i^{3/2}}, sgn(F_i)\right\}$$

  where I_i and O_i are the number of input and output words for processor i, respectively

- If we assume that the serial lower bounds apply for all processors, we obtain a general lower bound for program runtime:

Theorem (General Heterogeneous Lower Bound)

For P processors and G total flops to execute, then

$$T \geq \min_{\sum F_i = G} \max_{1 \leq i \leq P} \left\{\gamma_i F_i + \beta_i \max\left\{I_i + O_i, \frac{F_i}{8\sqrt{M_i}}\right\} + \alpha_i \max\left\{\frac{I_i + O_i}{M_i}, \frac{F_i}{8M_i^{3/2}}, sgn(F_i)\right\}\right\}.$$

Heterogeneous Lower Bounds : matrix-vector operations

- In the case of matrix-vector operations $I + O = O(n^2) = G$, so

$$T \geq \min_{\sum F_i = G} \left\{ \max_{1 \leq i \leq P} \left\{ \gamma_i F_i + \beta_i (cF_i) + \alpha_i \left(\frac{cF_i}{M_i} \right) \right\} \right\}$$

  with c constant

- We can solve an associated linear program (realizing that all processors must finish at the same time in the minimal partition) to obtain the optimal partition of flops:

$$F_i = \frac{\frac{1}{\xi_i}}{\sum_j \frac{1}{\xi_j}} G$$

  where $\xi_i = \gamma_i + c\beta_i + \frac{c\alpha_i}{M_i}$

Heterogeneous Lower Bounds : matrix-matrix operations

- If we assume that $\frac{F_i}{8\sqrt{M_i}} > I_i + O_i$ (as in the case of matrix-matrix operations) we obtain an optimal partition in a similar manner:

$$F_i = \frac{\frac{1}{\delta_i}}{\sum_j \frac{1}{\delta_j}} G$$

where $\delta_i = \gamma_i + c\frac{\beta_i}{8\sqrt{M_i}} + \frac{c\alpha_i}{8M_i^{3/2}}$

Heterogeneous Algorithms : Challenges and Considerations

- If we attempt to use the model for work partitioning, we must accurately measure performance parameters...

Heterogeneous Algorithms : Challenges and Considerations

- If we attempt to use the model for work partitioning, we must accurately measure performance parameters...
- How to best measure: bandwidth, latency, etc.?

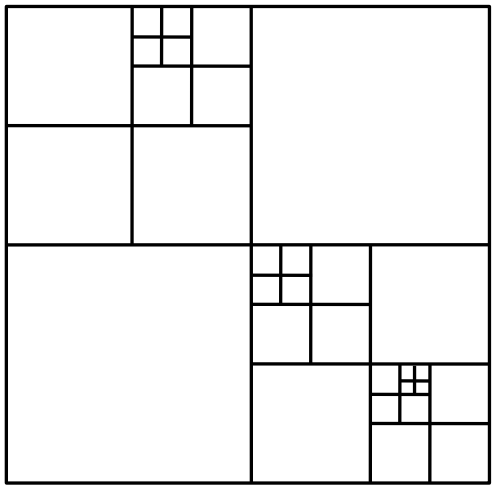
Heterogeneous Algorithms : Challenges and Considerations

- If we attempt to use the model for work partitioning, we must accurately measure performance parameters...
- How to best measure: bandwidth, latency, etc.?
- If the input data lies in global memory, how can we achieve latency goals?

Heterogeneous Algorithms : Challenges and Considerations

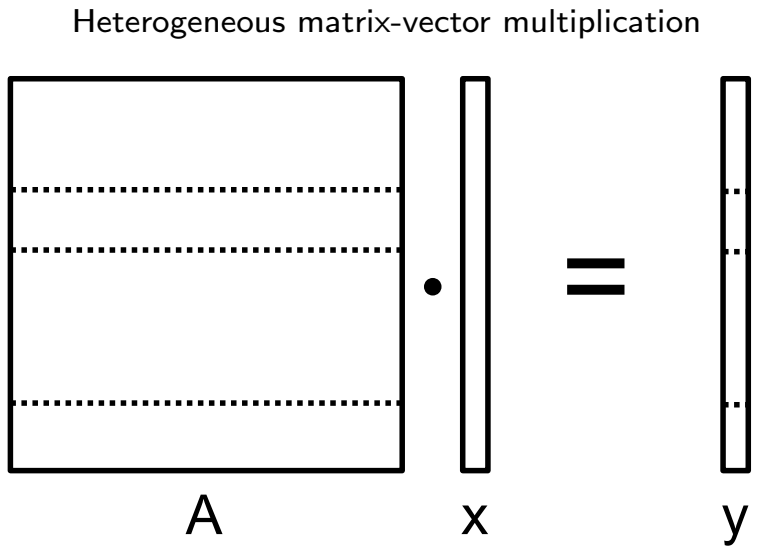
- If we attempt to use the model for work partitioning, we must accurately measure performance parameters...
- How to best measure: bandwidth, latency, etc.?
- If the input data lies in global memory, how can we achieve latency goals?
- Proper data layout should be considered to properly minimize the number of messages (latency)

Quadtree



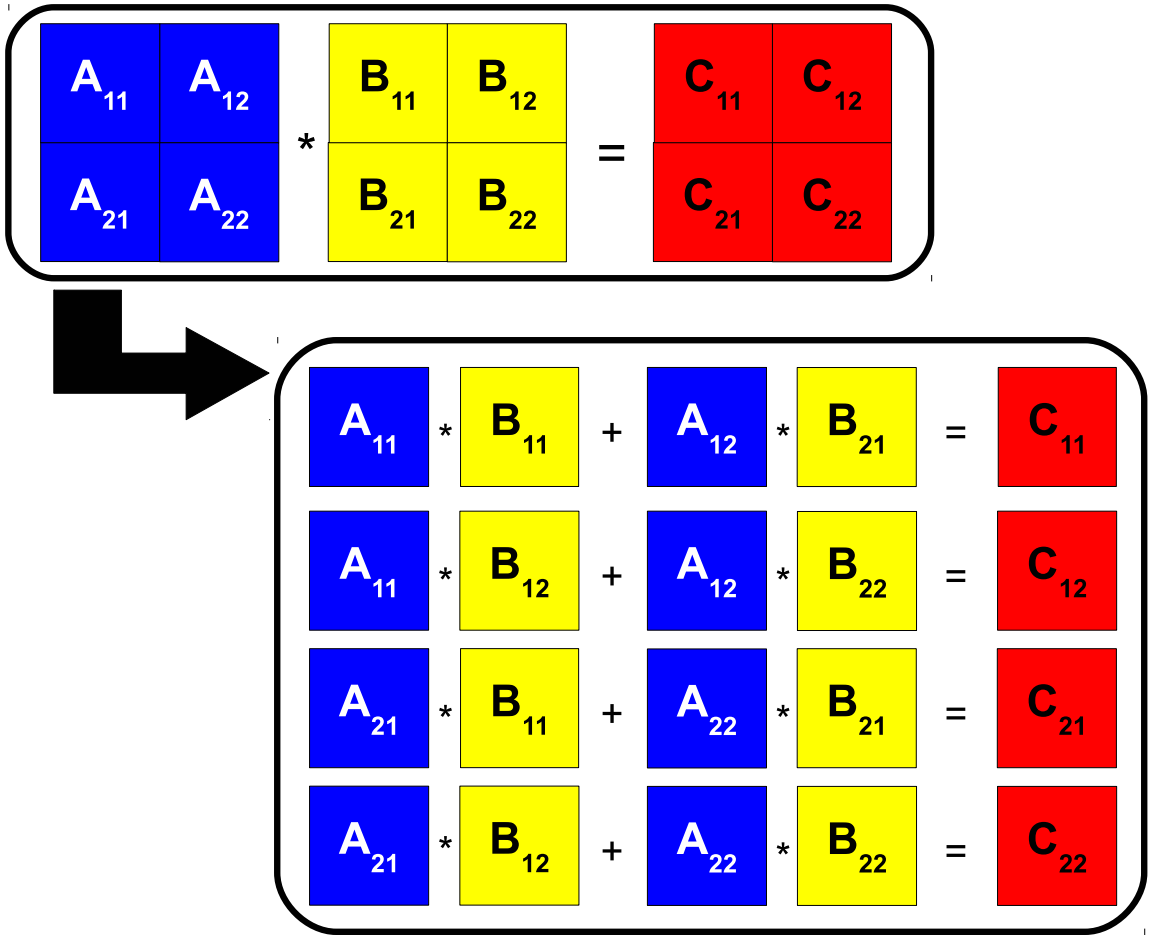
Heterogeneous Algorithms : Matrix-vector multiplication

- Algorithm Outline
  1. Determine parameters $\alpha_i, \beta_j, \gamma_i, M_i$ and use to calculate each F_i/G
  2. Split the input matrix (stored row-major) according to the values of F_i/G
  3. Each processor uses a fast matrix-vector multiplication routine to calculate its assigned work
  4. Merge results into output vector in global memory



Heterogeneous Algorithms : Matrix-matrix multiplication

- We can use recursive matrix multiplication to divide the work into 8 subproblems at each level of recursion



Heterogeneous Algorithms : Matrix-matrix multiplication

- We can use recursive matrix multiplication to divide the work into 8 subproblems at each level of recursion

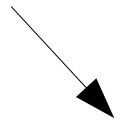
- Algorithm Overview
  1. Input matrices should be stored block-recusively to ensure contiguous subproblems
  2. Determine parameters $\alpha_i, \beta_j, \gamma_i, M_i$ and use to calculate each F_i/G
  3. Convert F_i/G to an octal fraction, and assign subproblems according to the octal digits
  4. Each processor computes work via a matrix multiplication routine tuned for M_i-sized problems
  5. Final output matrix is generated in global memory

Heterogeneous Algorithms : Matrix-matrix multiplication

	P1	P2	P3	P4
Decimal	.25	.25	.40625	.09375
Octal	.20	.20	.32	.06

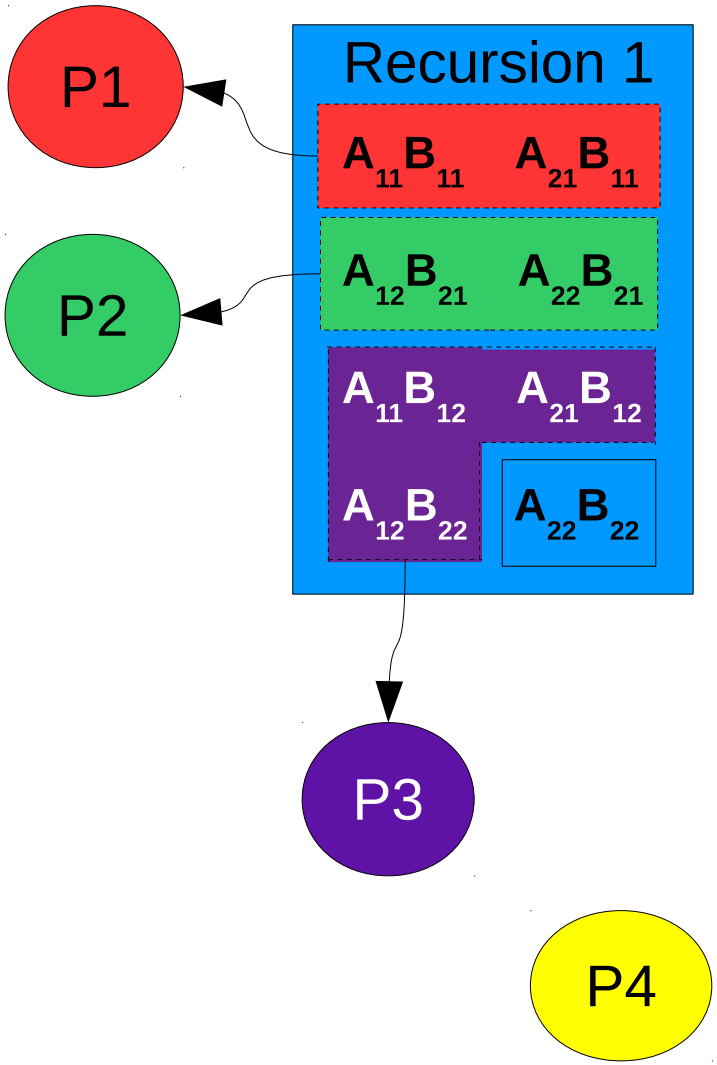
Heterogeneous Algorithms : Matrix-matrix multiplication

	P1	P2	P3	P4
Decimal	.25	.25	.40625	.09375
Octal	.20	.20	.32	.06



	P1	P2	P3	P4
Recursion 1	2	2	3	0
Recursion 2	0	0	2	6

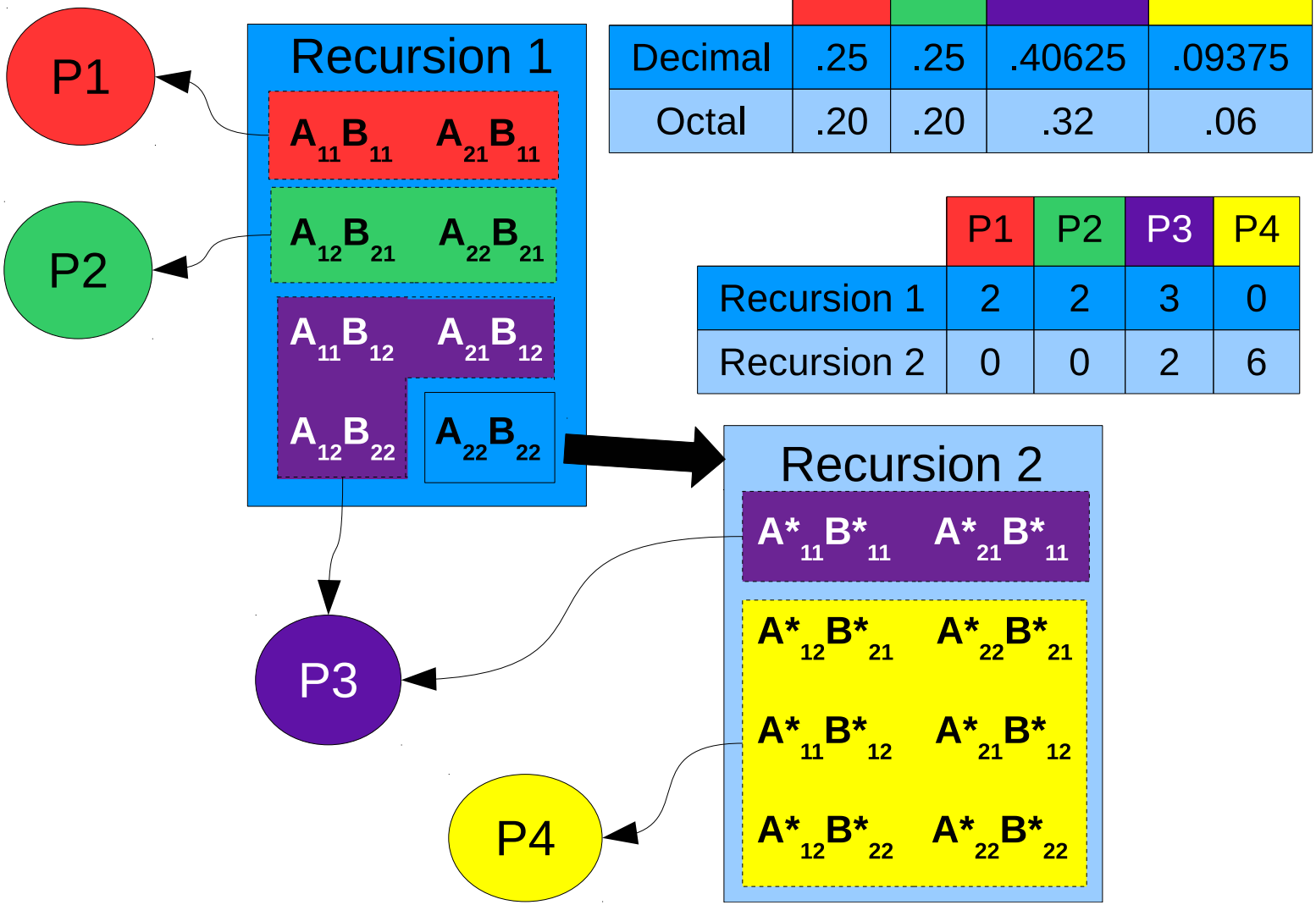
Heterogeneous Algorithms : Matrix-matrix multiplication



	P1	P2	P3	P4
Decimal	.25	.25	.40625	.09375
Octal	.20	.20	.32	.06

	P1	P2	P3	P4
Recursion 1	2	2	3	0
Recursion 2	0	0	2	6

Heterogeneous Algorithms : Matrix-matrix multiplication



	P1	P2	P3	P4
Decimal	.25	.25	.40625	.09375
Octal	.20	.20	.32	.06

	P1	P2	P3	P4
Recursion 1	2	2	3	0
Recursion 2	0	0	2	6

Open Questions

- Do we really have to implement a block-recursive data structure to achieve the latency bound, or can we get away with row-major?
- When can we simply ignore a processor? (i.e. cheaper to do small problems only on CPU, as opposed to CPU/GPU split)
- How big do problems have to be before we start to see benefit from a CA approach?
- What emerging architectures should be targeted for quantitative evaluation?

Future Work

- Implementation and evaluation of matrix-vector and matrix-matrix multiplication algorithms
- Extension of bounds to Strassen-like algorithms
- Algorithms for more complicated algorithms: LU and QR
  - Must consider flops/byte mapping and the critical path of the algorithm
- Can a dynamic scheduling/work stealing approach provide more flexibility? (handling heterogeneity in time as well as space)
- What would an energy-optimal algorithm look like on a heterogeneous machine? How does this relate to the CA paradigm?