Modeling the Benefits of Mixed Data and Task Parallelism

Soumen Chakrabarti

U. C. Berkeley and IBM

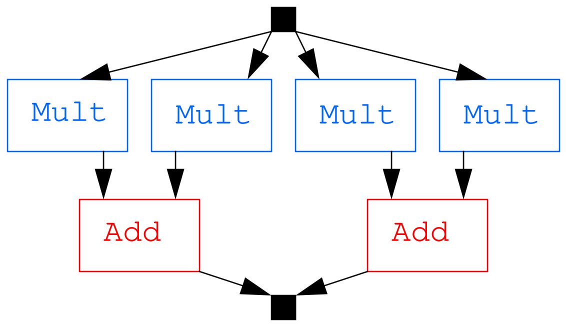
Joint work with

Jim Demmel

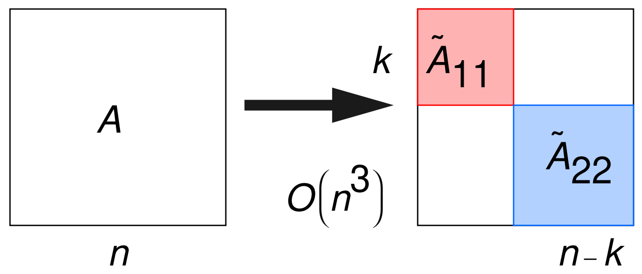
Kathy Yelick

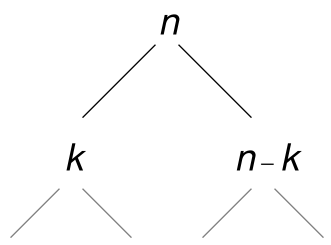
What is mixed parallelism?

Example 1: Complex matrix multiplication



Example 2: Nonsymmetric eigenvalue algorithm





Generalize:

- A DAG of tasks and precedence edges
- Task parallelism as permitted by precedence
- Data parallelism within each "malleable" task

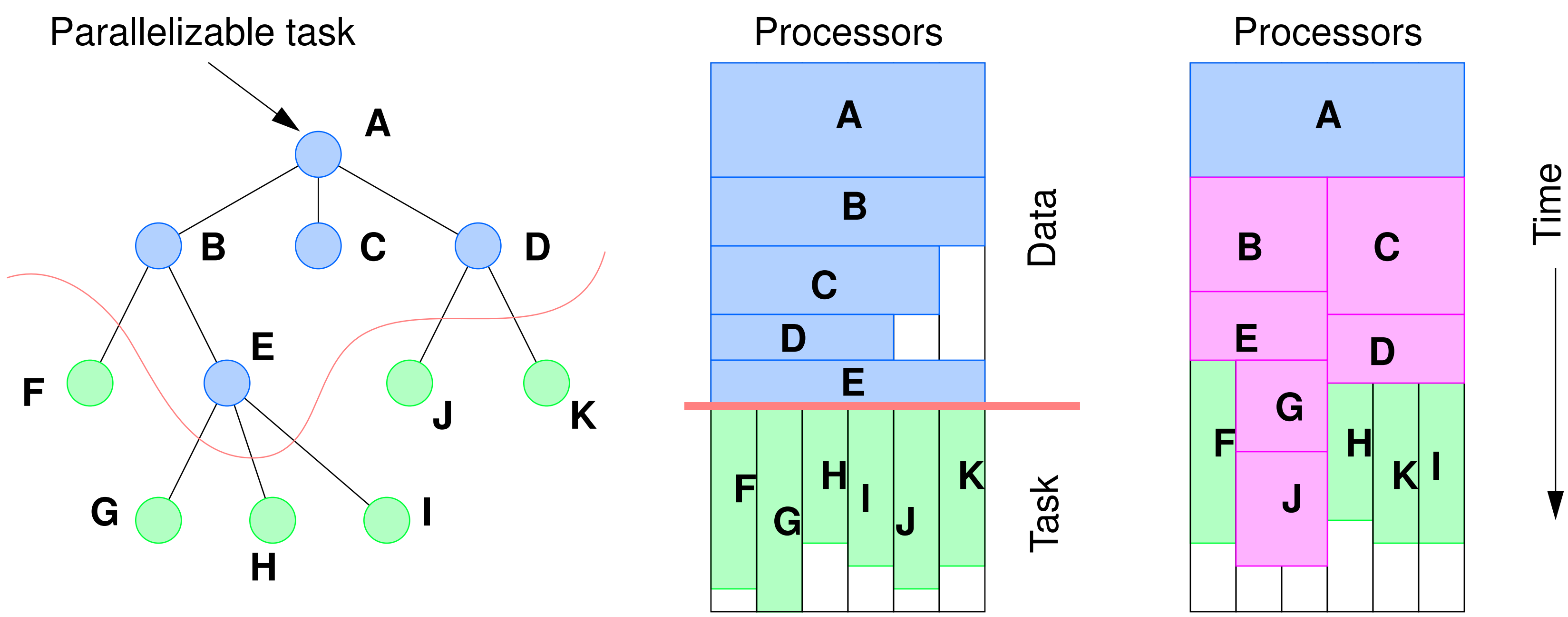
Execution strategies

Pure data parallelism: each task occupies all *P* processors

Pure task parallelism: packing of sequential tasks into *P* bins

Combinations

- Switched
- General mixed



Task graph Switched General mixed

Outline

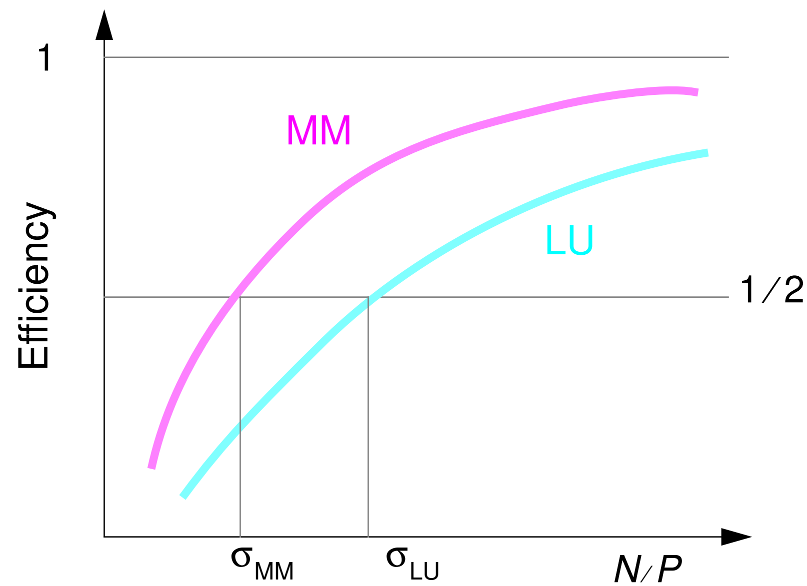
Goals of this talk:

- Design simple hybrid execution strategies
  - easier to implement
  - smaller runtime overhead
- Propose a performance model to compare these strategies
  - what problem and machine parameters to use
- Analyze performance gap between strategies using this model
  - simple may be close to best possible
- Validate using experimental data

Single task model

Variables: problem size N, number of processors P

Efficiency $e(N, P)$ increases with N, decreases with P



Example function: $e(N, P) = \dfrac{1}{1 + \sigma P / N}$

σ = size per processor for half of peak efficiency

Typical parameters

Trends in σ (half-efficiency point) and P (number of processors), e.g.: matrix multiply

	CM5	**Paragon**	**T3D**	**SP1**
σ_{MM}	53	633	1544	4250
P (typical)	256	128	128	64
$\sigma_{MM} P$ (typical)	14k	81k	200k	270k

Typical problem sizes (N):

- Very small, e.g., radar applications (FFT): $N = 32 \ldots 128$
- Very large: N in the millions

Typical degree of task parallelism (L):

- 4 in MM, 7-8 in CFD, 2-5 in weather modeling
- Much larger in dynamic task graphs

Batch model

- The instance:
  - L independent parallelizable tasks of sizes $N_1 \geq N_2 \geq \ldots \geq N_L$
  - i-th task needs uniprocessor time $f(N_i) = N_i^a$, $a > 1$
  - i-th task needs time $f(N_i) / (P \cdot e(N_i, P))$ using pure data parallelism
- The problem: For each i, schedule the i-th task on P processors (data parallel) or 1 processor (task parallel), to get minimum finish time
- Exact problem is intractable

Prefix-Suffix heuristic

- Sort the tasks $N_1 \geq N_2 \geq \ldots \geq N_L$
- Restrict decision to executing
  - a prefix of large tasks using data parallelism
  - the remaining suffix using task parallelism
- Compare all prefix choices in linear time

Performance: Within a factor of $2 + \varepsilon$ of best switched schedule possible, any $\varepsilon > 0$.

Note that sorting is useless if batch is balanced (all N_i equal).

Performance gap

e_D, e_S, e_M denote data, switched, and mixed parallel efficiency.

What choice of $N_1, \ldots, N_L$ minimizes e_D (data parallel efficiency)?

Possible invariants:

- constant total size $\sum N_i = N$
- constant total work $\sum f(N_i) = F$

Observations:

- For $f(x) = x^a$, $a > 1$, e_D is minimized for all N_i equal
- The ratio e_M / e_D is bounded above by

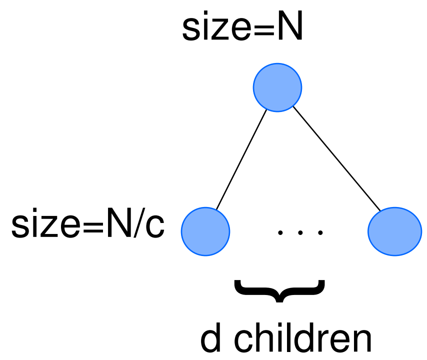
$$\frac{e_M}{e_S} \leq \frac{e_M}{e_D} \leq 1 + \frac{\sigma PL}{N}$$

- If $N \gg \sigma PL$, data parallelism is good enough
- Switched parallelism is no worse

Regular tree model

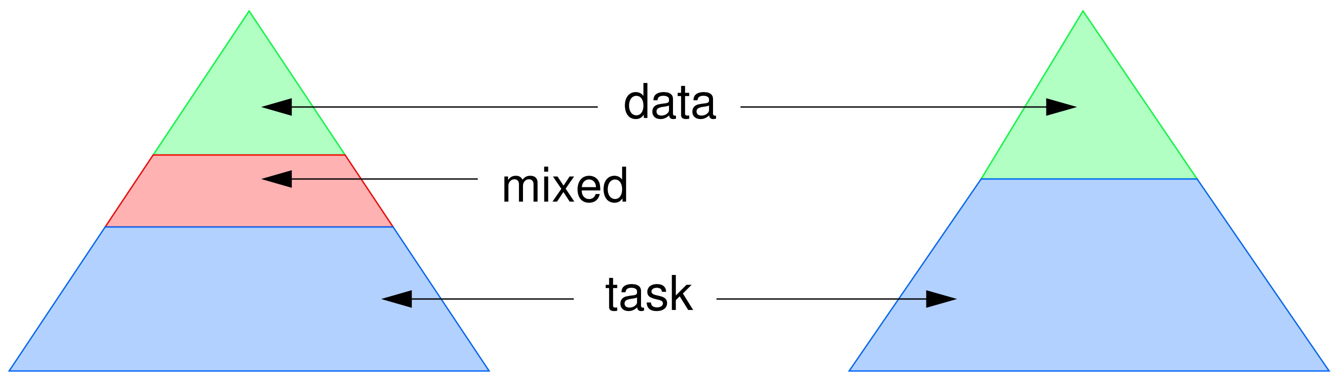
Extend balanced batch to form a complete balanced tree

- Root has size N, $d > 1$ children each of size N/c, $c > 1$, and so on
- Only data parallelism available at root
- Task parallelism is ideal after $\log_d P$ levels



Intuition:

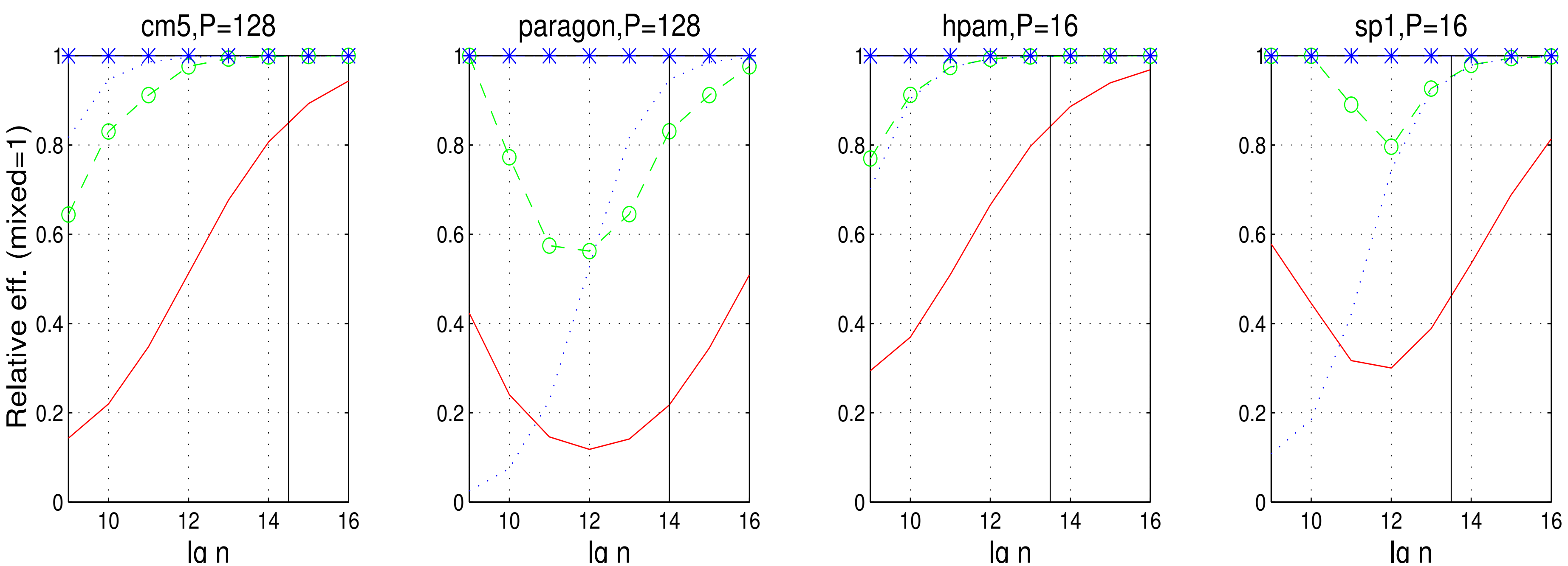
- Large nodes near root run well with pure data parallelism
- Many nodes generated by the time tasks get small



Efficiency improvements

Sparse Cholesky (fixed *P*)

data---- switched--- mixed ---- memory limit |

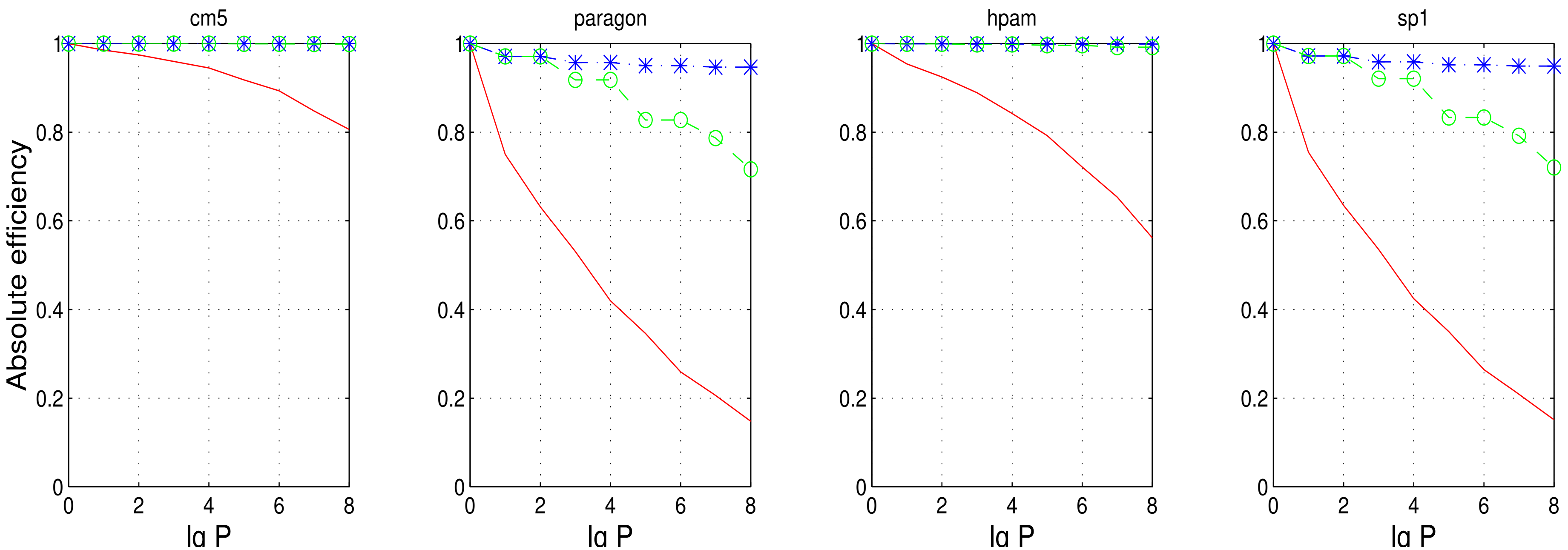


problem size

Scalability improvements

Sparse Cholesky (problem large enough to fill memory)

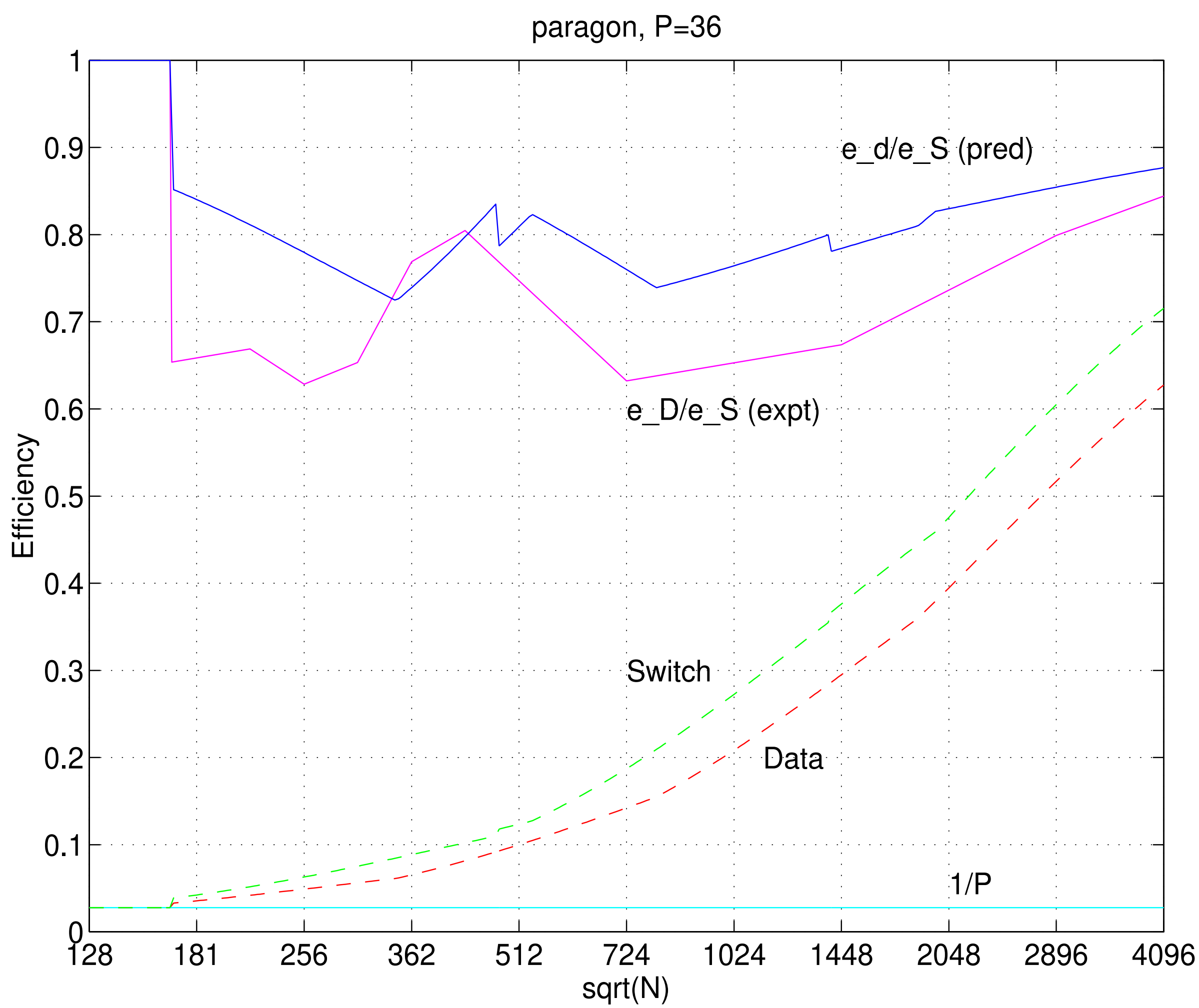
data---- switched--- mixed ----



machine size

Experimental validation

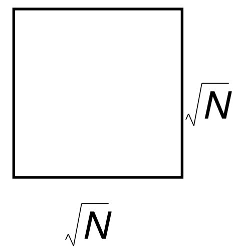
Switched vs data parallel eigenvalue program[1]



e = efficiency

D = data parallel

S = switched parallel



1. Courtesy: ORNL Paragon

Related work

Reference	Control Dependency	Task-data relation	Notes
Belkhale et al	Dynamic	No data	2.62-competitive online
Feldmann et al	Dynamic	No data	Lower bounds 0/1 efficiency profile
Ludwig et al, Turek at al	None	No data	Minimize finish or flow time. Offline.
Gross et al	Static		iWarp system. Small FFT from DSP applications
Ramaswamy et al	Static	Data transfer cost is modeled	Convex programming formulation
Foster et al, Chapman et al	Static		Software support to integrate data and task parallel Fortran

Concluding remarks

- In this paper we have:
  - studied various styles of hybrid parallel execution
  - modeled, predicted, validated, and interpreted performance
- Loose ends:
  - general mixed parallel efficiency is being over-estimated
  - need to analyze irregular task graph heuristics
- How *not* to find mixed parallel applications:
  - typical problem size is moderate to large
  - task graph is static with small width
  - can efficiently run only small or large tasks at a given time