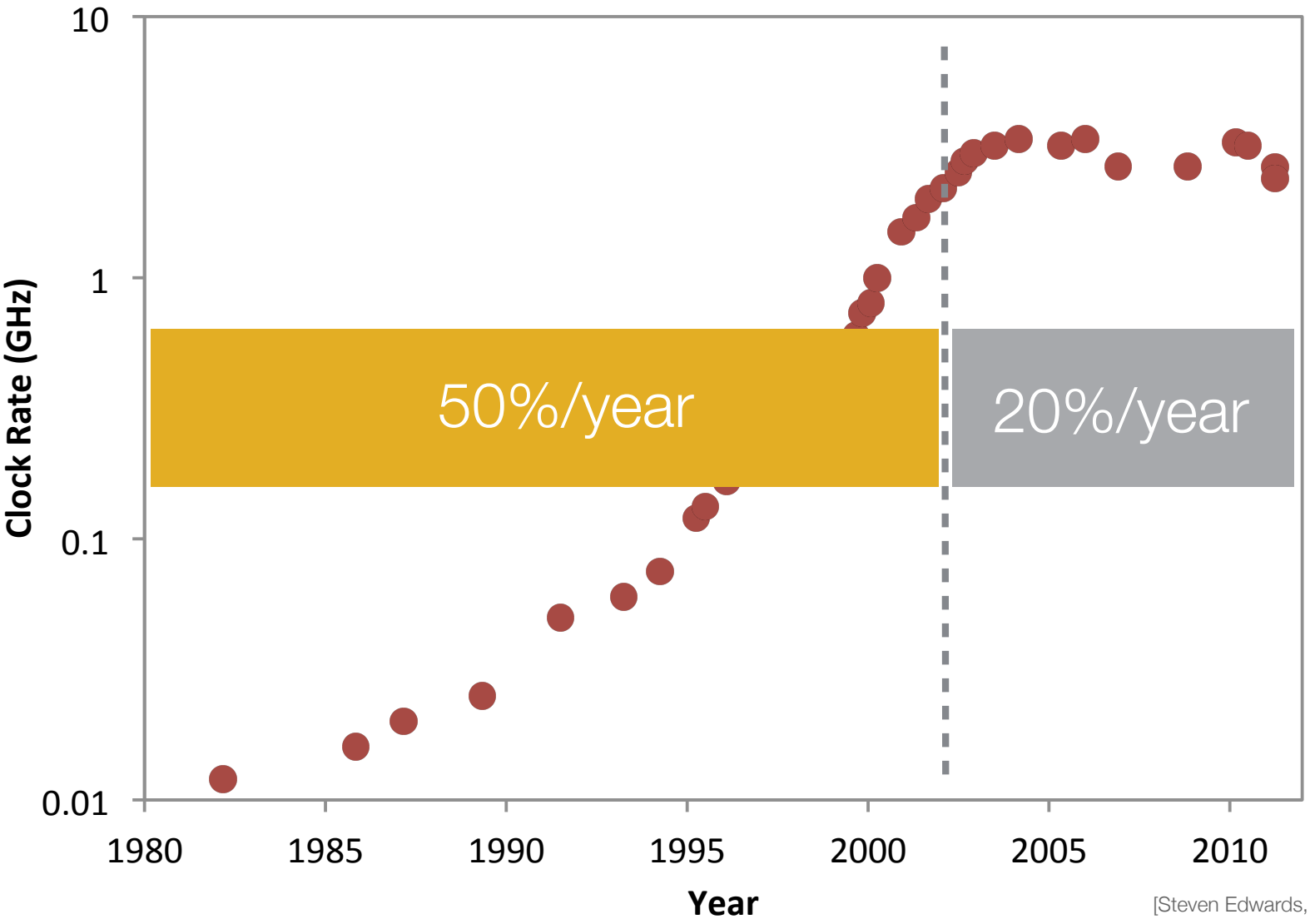
# Parallel Architectures and Algorithms for Large-Scale Nonlinear Programming

Carl D. Laird
Associate Professor, School of Chemical Engineering, Purdue University
Faculty Fellow, Mary Kay O'Connor Process Safety Center

Laird Research Group: http://allthingsoptimal.com

PURDUE CHEMICAL ENGINEERING

# Landscape of Scientific Computing

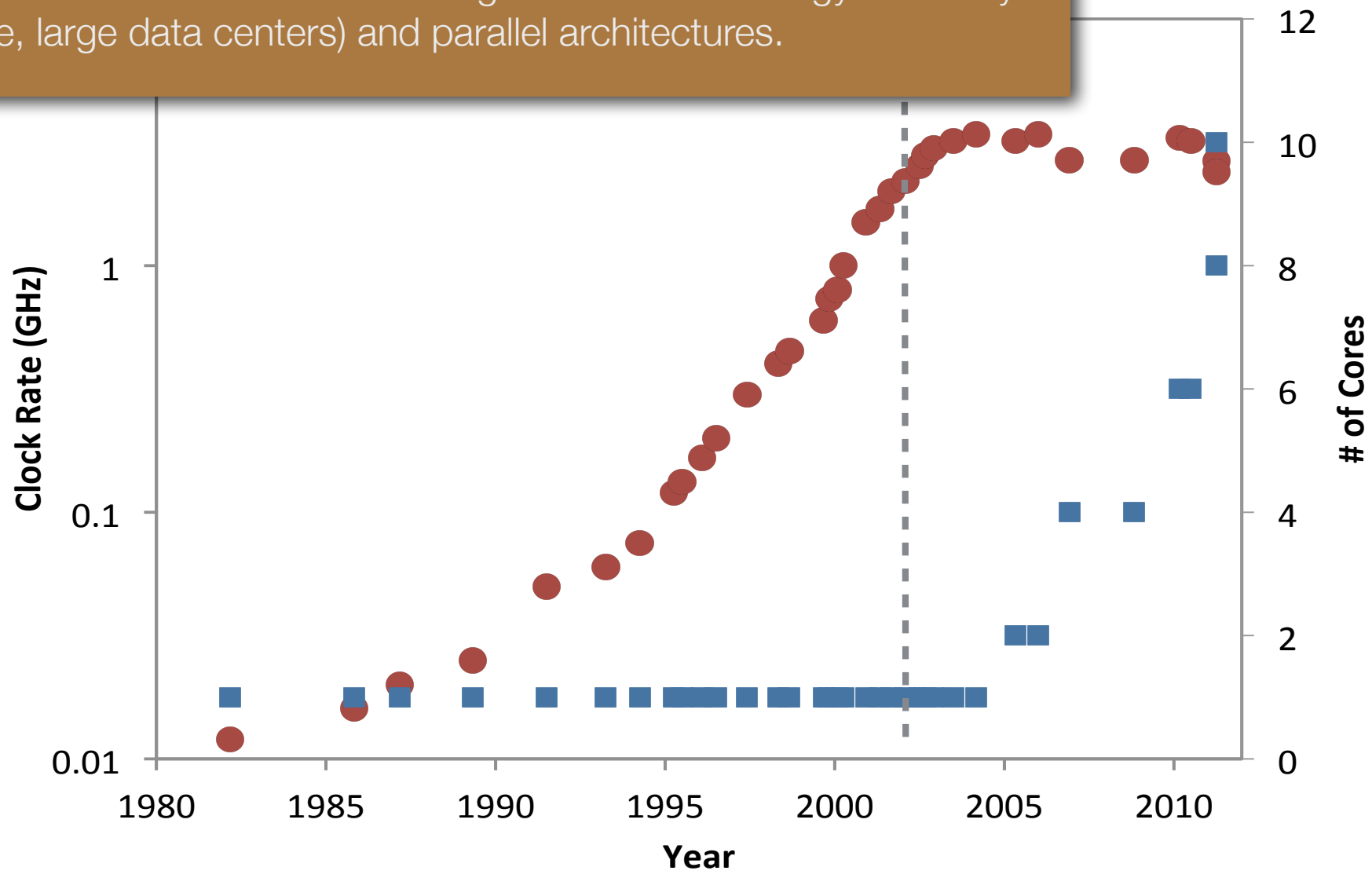



[Steven Edwards, Columbia University]

# Landscape of Scientific Computing

Clock-rate, the source of past speed improvements have stagnated.

Hardware manufacturers are shifting their focus to energy efficiency (mobile, large data centers) and parallel architectures.



Frontiers in Chemical Engineering: Research Needs and Opportunities (1988)


Panel on Computer Assisted Process and Control Engineering

ARTHUR W. WESTERBERG (*Chairman*), Carnegie Mellon University
HENRY CHIEN, Monsanto Corporation
JAMES M. DOUGLAS, University of Massachusetts
BRUCE A. FINLAYSON, University of Washington
ROLAND KEUNINGS, University of California, Berkeley
MANFRED MORARI, California Institute of Technology
JEFFREY J. SIIROLA, Eastman Kodak Company
WILLIAM SILLIMAN, Exxon Production Research Company

## USING THE COMPUTER'S POTENTIAL

Each decade over the last 35 years has seen the processing speed of newly designed computers increase by a factor of about 100 owing to advances in the design of electronic microcircuits and other computer hardware. On top of this has been another 100-fold increase per decade in computer speed, thanks to more efficient methods of carrying out computations (algorithms). It is not widely appreciated that new algorithms have been as valuable as hardware design in improving computer performance. With the combination of improved computer hardware and better algorithms, effective computer speeds have more than doubled on average each year.

"… over a 15-year span… [problem] calculations improved by a factor of 43 million. [A] factor of roughly 1,000 was attributable to faster processor speeds, … [yet] a factor of 43,000 was due to improvements in the efficiency of software algorithms."
- attributed to Martin Grotschel

[Steve Lohr, "Software Progress Beats Moore's Law", New York Times, March 7, 2011]

# Landscape of Computing

# Landscape of Computing

## High-Performance Parallel Architectures

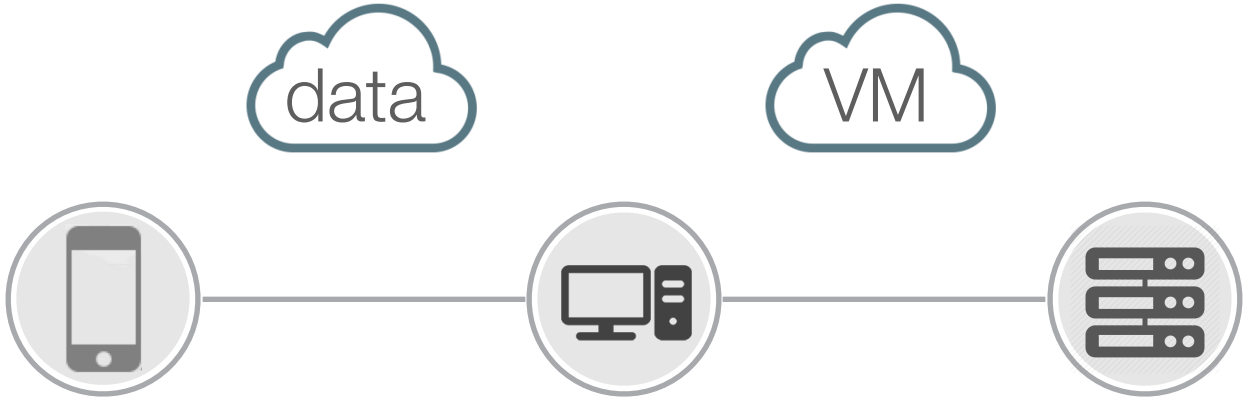
Multi-core, Grid, HPC clusters, Specialized Architectures (GPU)

# Landscape of Computing

High-Performance
Parallel Architectures

Multi-core, Grid,
HPC clusters,
Specialized
Architectures (GPU)

# Landscape of Computing




iOS and Android device sales have surpassed PC

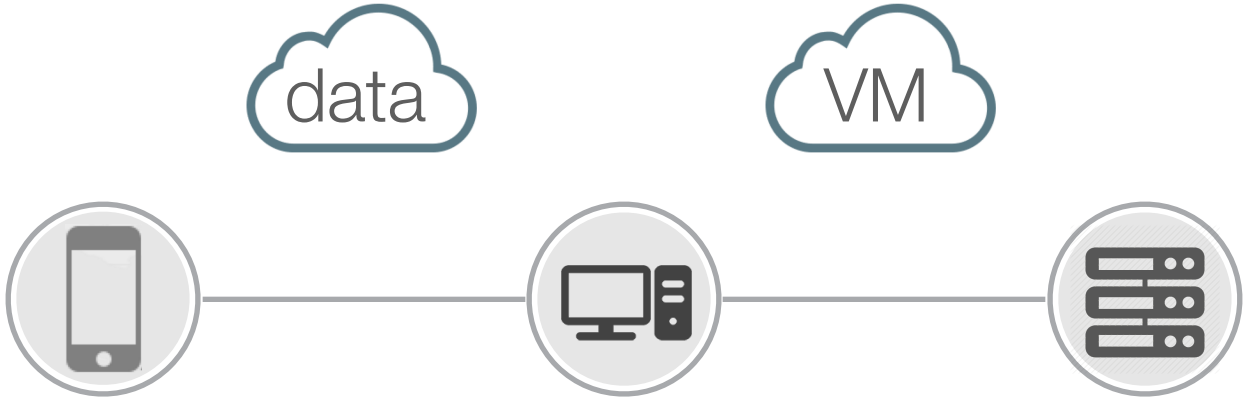
iPhone performance ~2X / year

iPhone 5S within a factor of 2 of 2.5 GHz Core i5 [Geekbench 3]

High-Performance Parallel Architectures

Multi-core, Grid, HPC clusters, Specialized Architectures (GPU)

# Landscape of Computing




Watch: increased use in operations

Watch: rise of medical devices and health

Watch: touch form factor for desktop computing

High-Performance Parallel Architectures

Multi-core, Grid, HPC clusters, Specialized Architectures (GPU)

# High-Performance Computing

Parallel Computing Architectures

Measuring Parallel Performance

Parallel Algorithms for NLP

# Parallel Computing Architectures

## Flynn's Taxonomy

| | Single Data | Multiple Data |
|---|---|---|
| **Single Instruction** | SISD | SIMD |
| **Multiple Instruction** | MISD | MIMD |

# Parallel Computing Architectures

## Flynn's Taxonomy

| | Single Data | Multiple Data |
|---|---|---|
| **Single Instruction** | SISD | SIMD |
| **Multiple Instruction** | MISD | MIMD |

**Beowulf Cluster (MIMD)**

- Scalable Distributed Computing
  100s of Processors
- Standard Compilers (MPI)
- Network Communication (Ethernet)
- Communication Bottleneck

**Desktop Multi-core (MIMD)**

- Affordable Hardware
  Low Number of Processors
- Standard Compilers (Threads, OpenMP)
- Fast Communication (No Network)
- Memory Access/# CPU Bottleneck

# Parallel Computing Architectures

## Flynn's Taxonomy

| | Single Data | Multiple Data |
|---|---|---|
| **Single Instruction** | SISD | SIMD |
| **Multiple Instruction** | MISD | MIMD |

**Emerging Architectures (SIMD/Hybrid)**

Graphics Processing Unit (GPU)

- Affordable, 1000's of Processors
- Specialized Compilers/Languages
  - CUDA, Brook++, OpenCL
- Several Complexities & Limitations

**Beowulf Cluster (MIMD)**

- Scalable Distributed Computing
  100s of Processors
- Standard Compilers (MPI)
- Network Communication (Ethernet)
- Communication Bottleneck

**Desktop Multi-core (MIMD)**

- Affordable Hardware
  Low Number of Processors
- Standard Compilers (Threads, OpenMP)
- Fast Communication (No Network)
- Memory Access/# CPU Bottleneck

# Writing High-Performance Scientific Software

- Implementation and hardware details matter
  - Cache sizes have a major impact
  - Memory latency, coalescing, and alignment matter
  - Network latency and bandwidth matter
  - Compiler optimization can only do so much
  - Use high-performance code tailored to architecture (e.g. BLAS & LAPACK for dense linear algebra)

- Languages matter
  - C / C++ / Fortran still the dominant choices
  - Interpreted languages (E.g. Matlab / Python) might be reasonable, but be careful
  - Modern compiled languages close to speed of C (with significant reduction in development time)

# Measuring Parallel Performance

Speedup: $$S = \frac{\text{Time for } \textbf{best known} \text{ serial algorithm}}{\text{Time for parallel algorithm}}$$

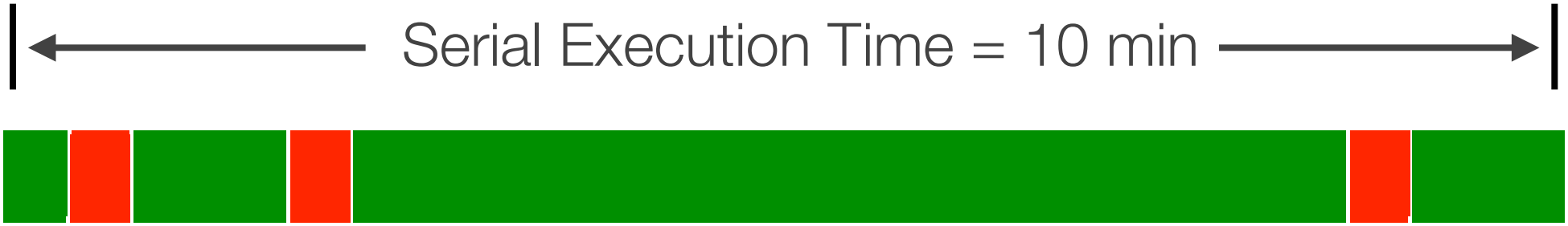
Efficiency: $$\frac{\text{Actual Speedup}}{\text{Ideal Speedup}} = \frac{S}{N_p}$$ usually < 1

Communication Overhead

Memory/Cache Bottlenecks

# Measuring Parallel Performance

# Measuring Parallel Performance




Can be computed in parallel

Must be computed in serial

# Measuring Parallel Performance


Can be computed in parallel

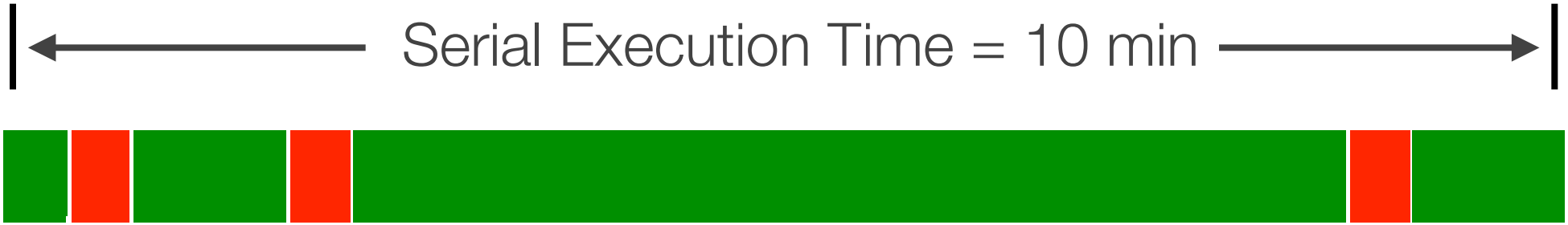
Must be computed in serial

# Measuring Parallel Performance


Can be computed in parallel

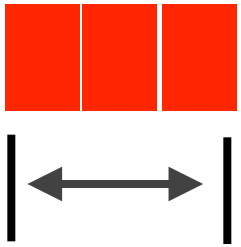
Must be computed in serial

# Measuring Parallel Performance





2 Processors
1.8 Times Speedup
~90% Efficiency

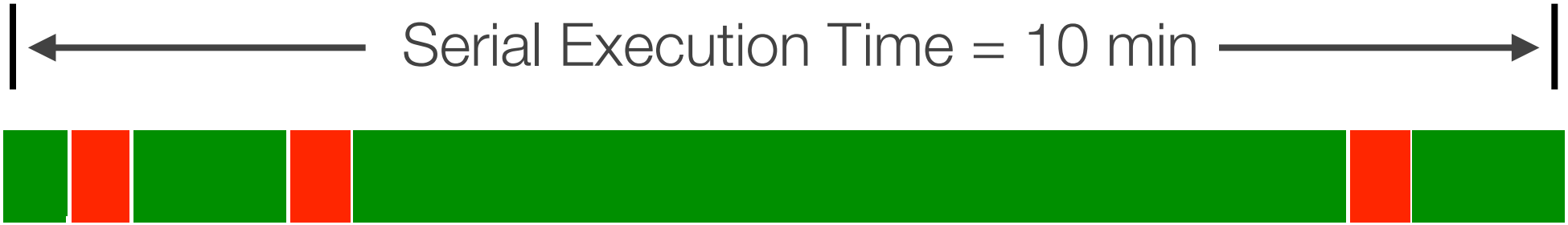



Infinite Processors
10 Times Speedup

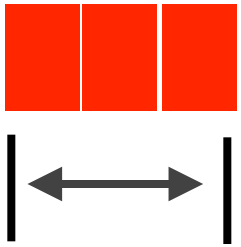
Can be computed in parallel

Must be computed in serial

# Measuring Parallel Performance




Amdahl's Law, Maximum Speedup: $S_\infty = \frac{1}{\phi_s}$



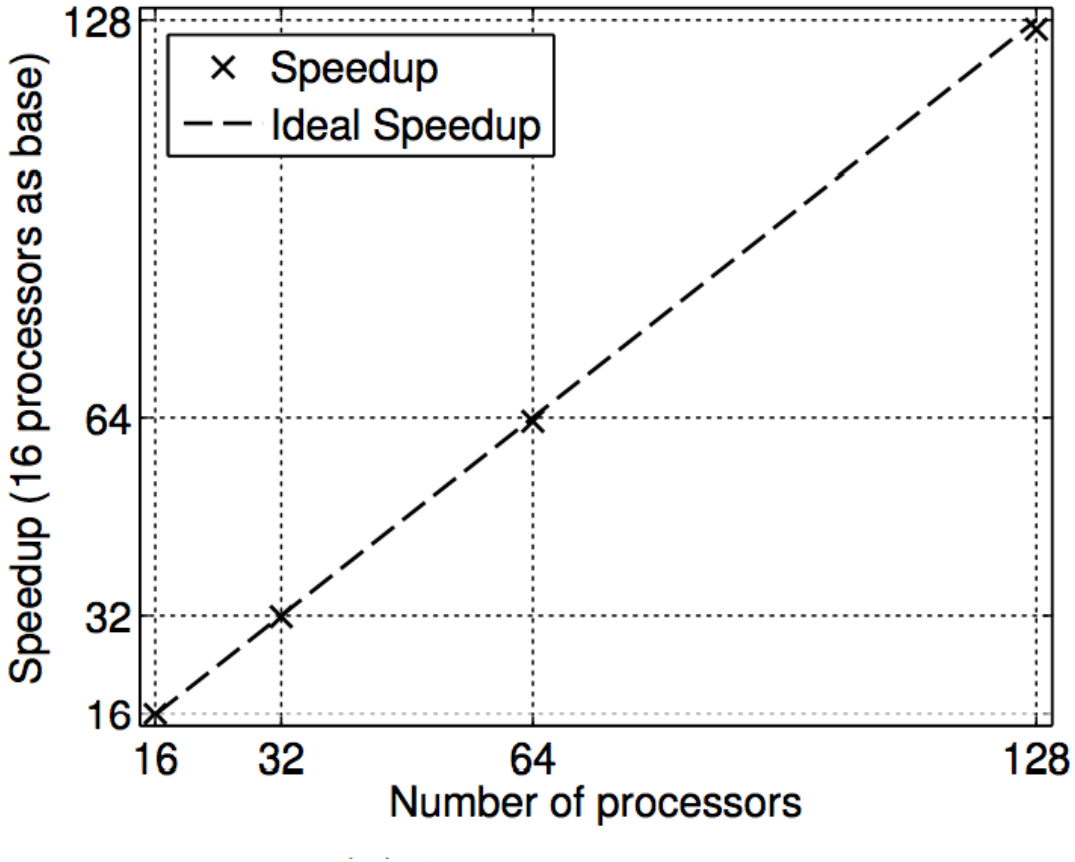
Parallel Time = 1.0 min

Infinite Processors
10 Times Speedup

Can be computed in parallel

Must be computed in serial

# Measuring Parallel Performance (Scaling Efficiency)

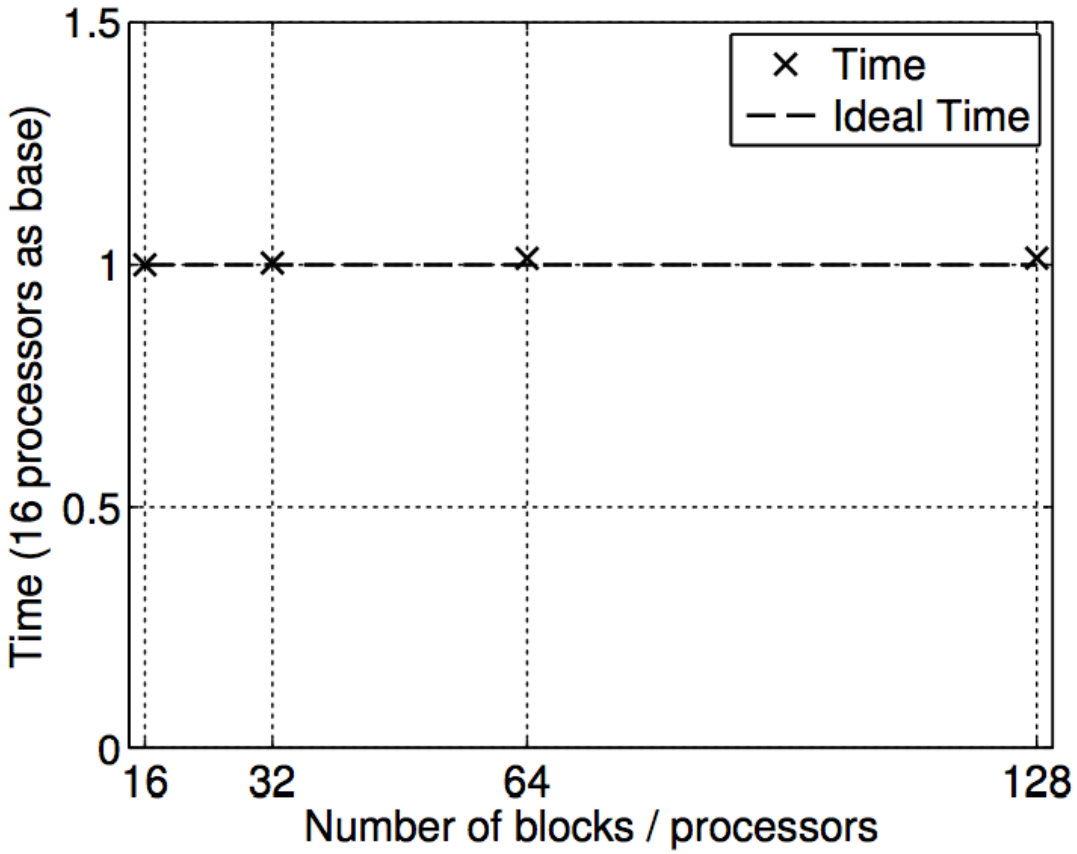



Strong Scaling
- Problem size fixed as # processors increased
- Fixed workload

Linear Scaling
- Speedup = # Processors

Performance compared against linear scaling

# Measuring Parallel Performance (Scaling Efficiency)

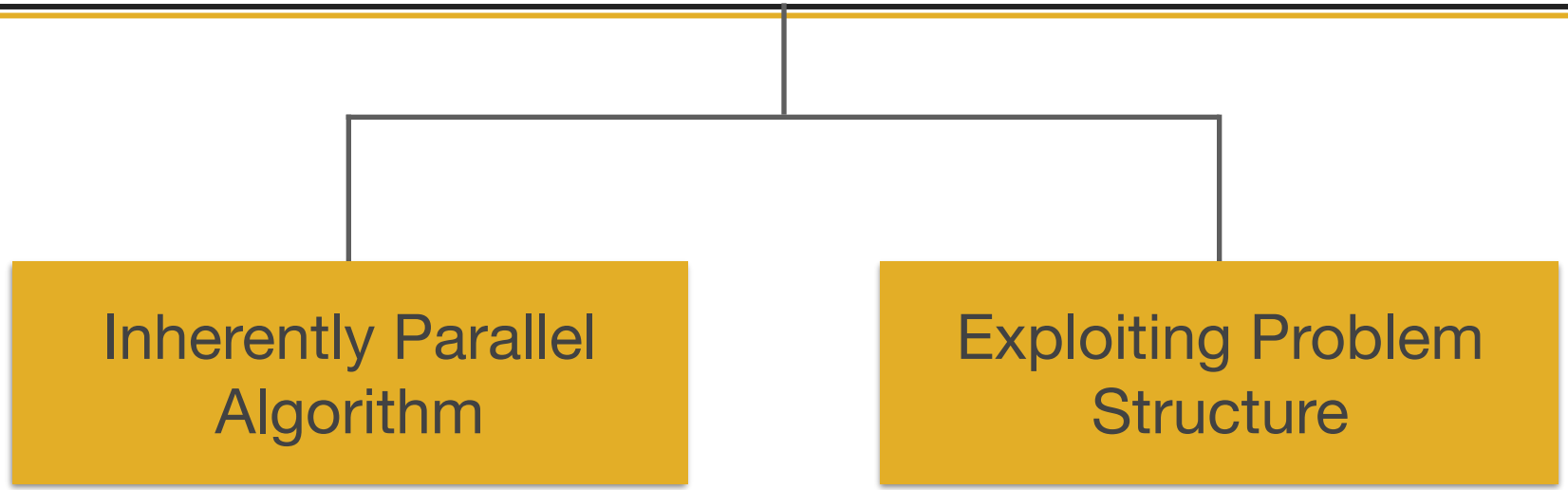



Weak Scaling
- Problem size grows as # processors increases
- Fixed workload per processor

Linear Scaling
- Constant run-time

Performance compared against linear scaling

# Parallel Algorithms for Nonlinear Optimization




Algorithm Parallel on General Problem

Often based on less effective serial algorithms

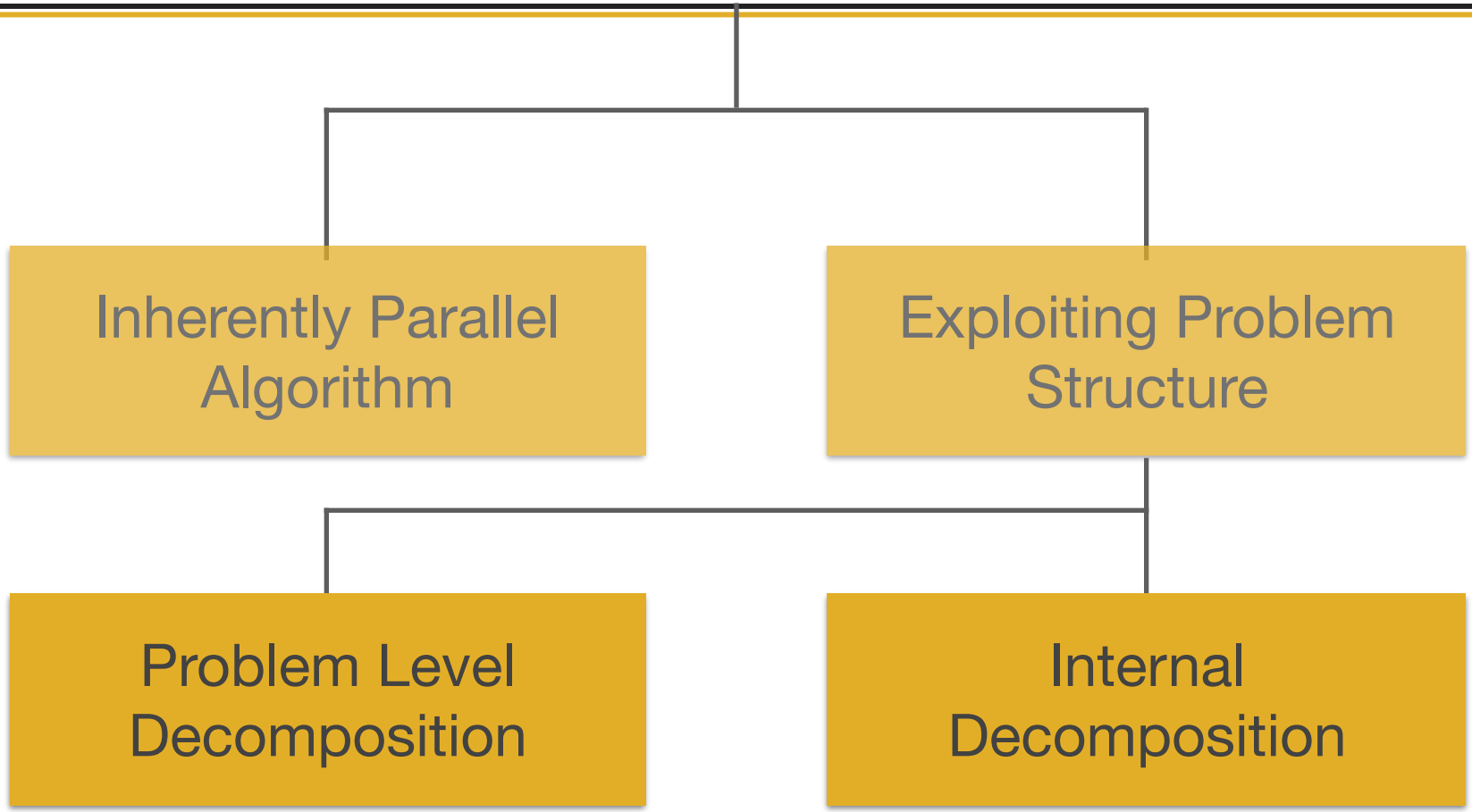
Easily parallelized

Algorithm Parallel Due to Problem Structure

Decomposition strategies

High-level parallelism

# Parallel Algorithms for Nonlinear Optimization



# Nonlinear Interior-point Methods

**Original NLP**

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

**Barrier NLP**

$$\begin{aligned} \min_{x} \quad & f(x) - \mu \cdot \sum_{i} ln(x_i) \\ \text{s.t.} \quad & c(x) = 0 \end{aligned}$$

as $x \to 0, \quad -ln(x) \to \infty$

as $\mu \to 0, \quad x^{\star}(\mu) \to x^{\star}$

Fiacco & McCormick (1968)

# Nonlinear Interior-point Methods

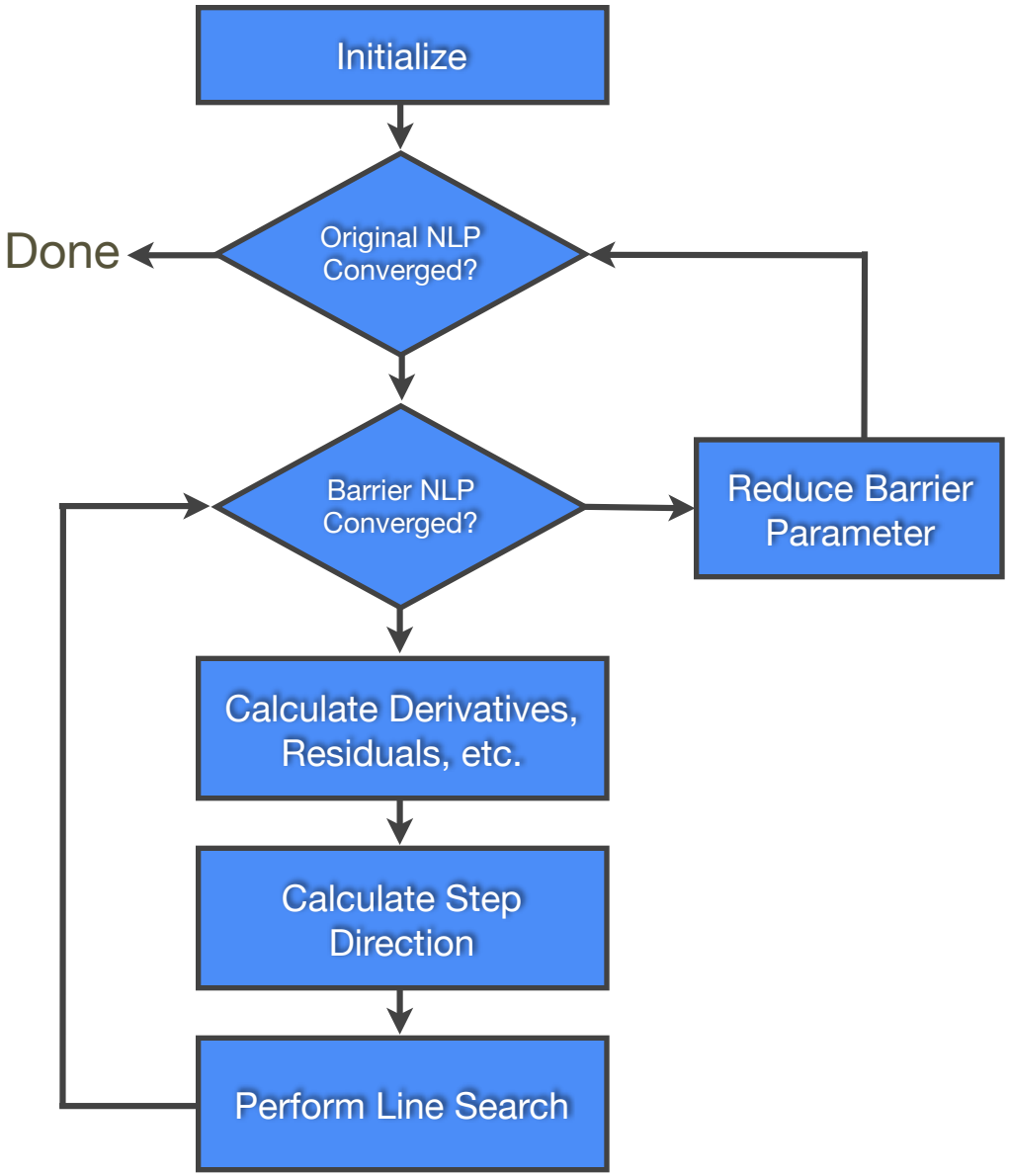
**Original NLP**

$$\min_x \quad f(x)$$
$$\text{s.t.} \quad c(x) = 0$$
$$x \geq 0$$

**Barrier NLP**

$$\min_x \quad f(x) - \mu \cdot \sum_i ln(x_i)$$
$$\text{s.t.} \quad c(x) = 0$$

KNITRO (Byrd, Nocedal, Hribar, Waltz)
LOQO (Benson, Vanderbei, Shanno)
IPOPT (Wachter, Laird, Biegler)



# Nonlinear Interior-point Methods

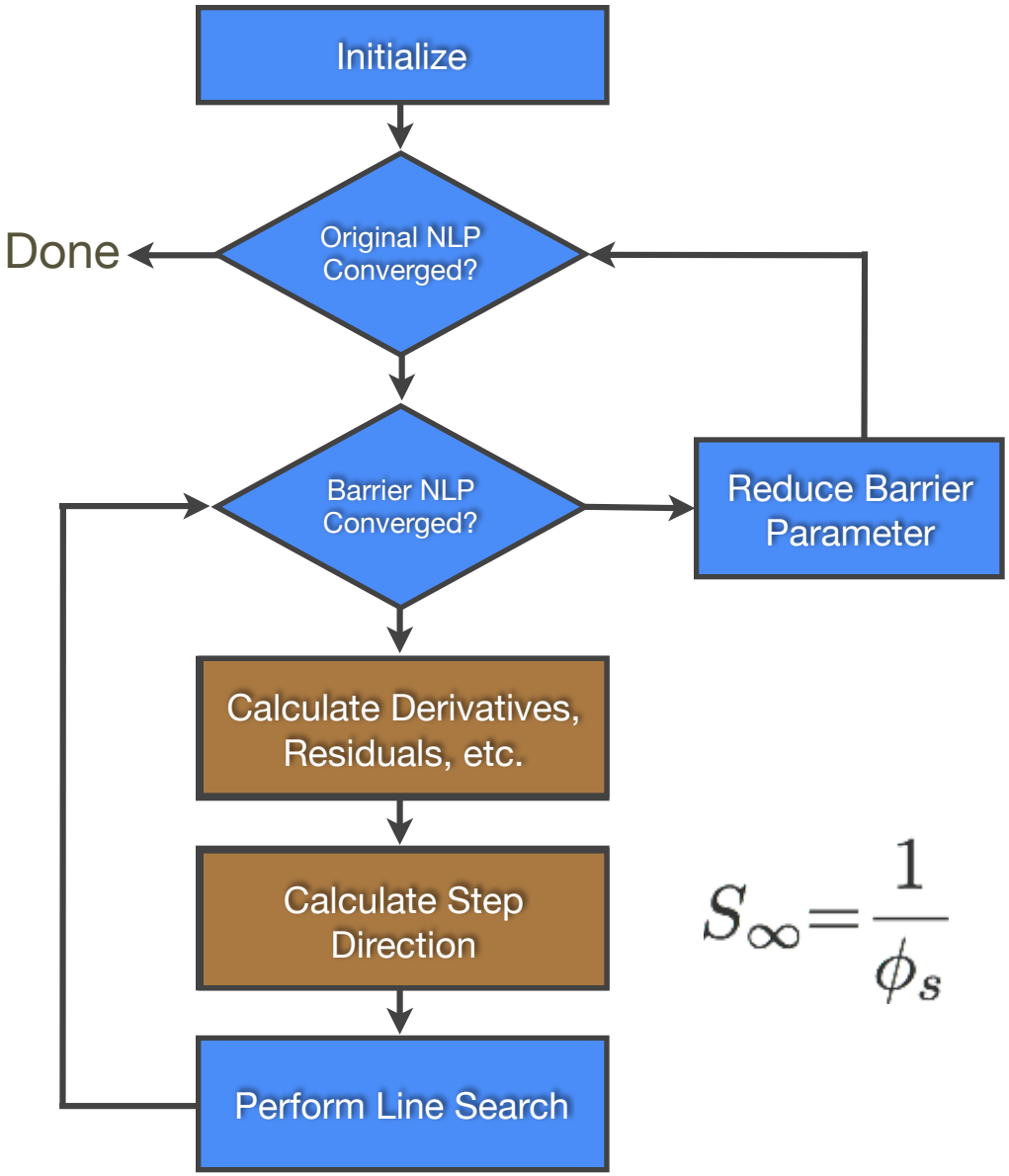
**Original NLP**

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

**Barrier NLP**

$$\begin{aligned} \min_x \quad & f(x) - \mu \cdot \sum_i ln(x_i) \\ \text{s.t.} \quad & c(x) = 0 \end{aligned}$$

KNITRO (Byrd, Nocedal, Hribar, Waltz)
LOQO (Benson, Vanderbei, Shanno)
IPOPT (Wachter, Laird, Biegler)




$$S_\infty = \frac{1}{\phi_s}$$

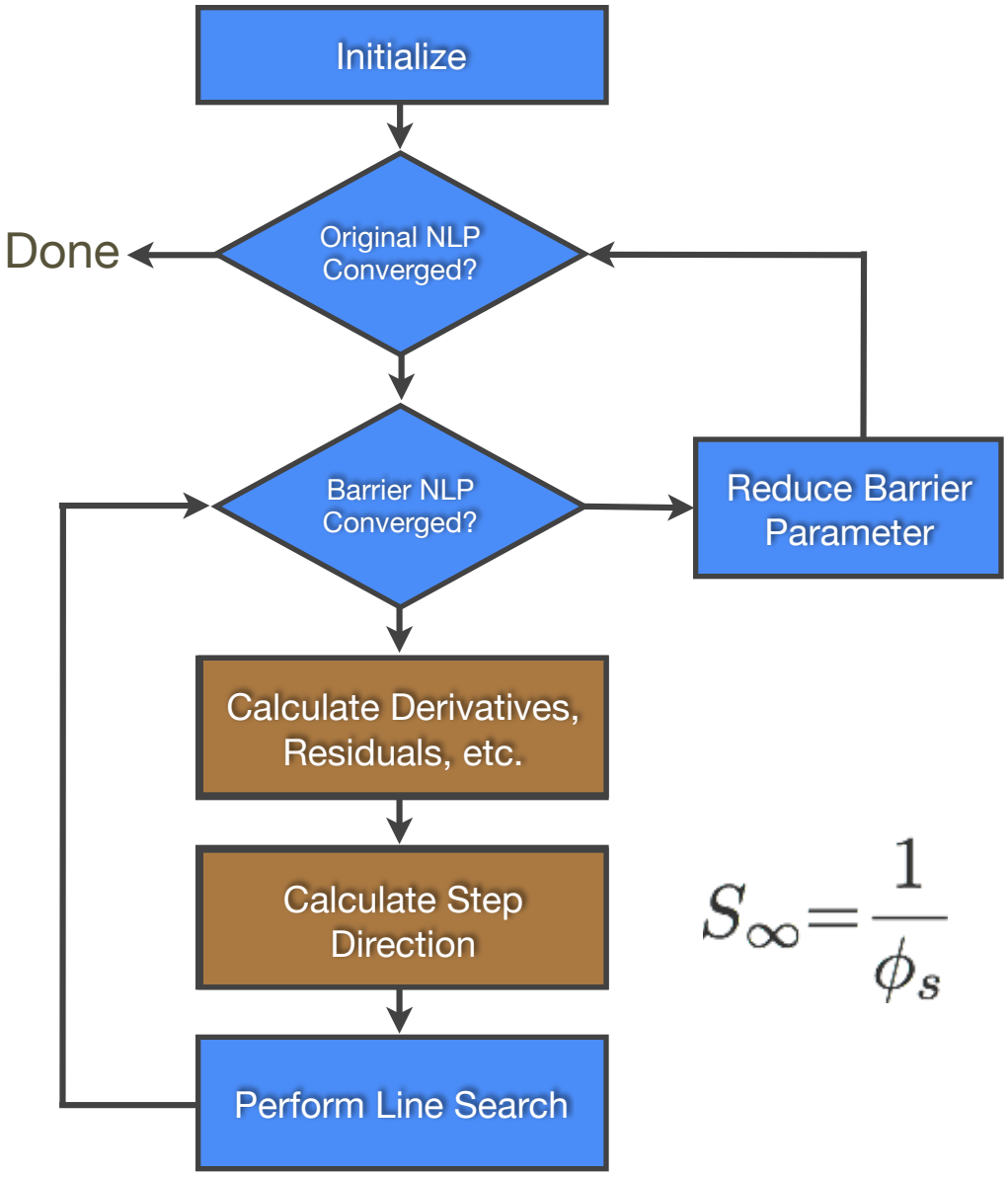
# Parallel Nonlinear Interior-point Methods

Structure in the optimization problem induces structure in the linear algebra

Parallelize all scale-dependent operations
- Vector and matrix operations
- Model evaluation

Compared with problem-level decomposition, implementation is time consuming

Retain convergence properties of serial algorithm




$$S_\infty = \frac{1}{\phi_s}$$

# Parallel Linear Algebra for NLP

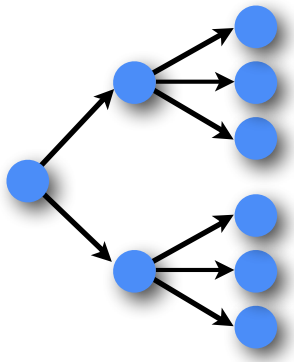
General Purpose Parallel Direct Solvers
- For general sparse systems - no need to specify structure
- [Schenk & Gartner 2004; Scott 2003; Amestoy et al. 2000; …]
- Modest scale up

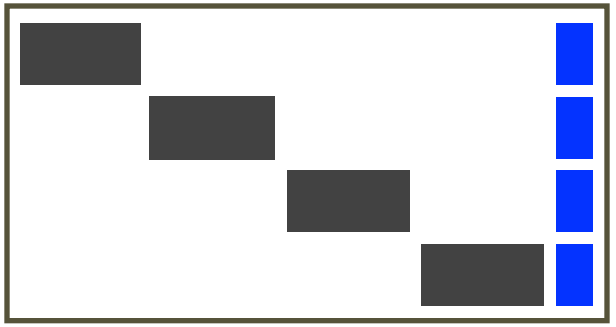
Iterative Linear Solvers (e.g. GMRES, PCG)
- Capable of handling general sparse systems
- Easily parallelized (many implementations)
- Appropriate for SIMD architectures
- Requires effective preconditioning
  [Dollar et al. 2007; Forsgren et al. 2008]

Tailored Decomposition of Problem Structure
- Custom parallel linear algebra exploits a fixed, known structure
- [Gondzio, Anitescu (PIPS), Laird, …]
- Problem class specific - difficult to implement
- Can scale to large parallel architectures
  (100's - 1000's of processors)

# Exploiting Problem Structure

Optimization Under Uncertainty
- block structure because of coupled scenarios
- common structure of many applications

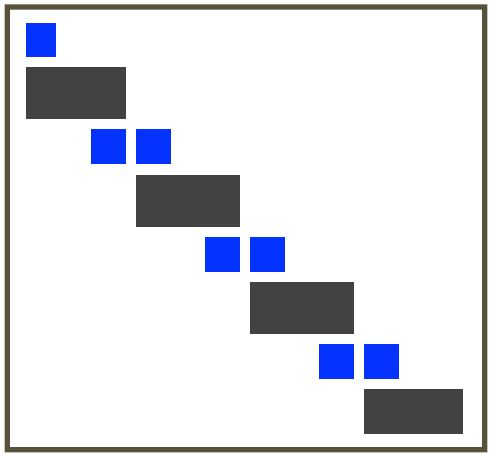




Kang, J., Word, D.P., and Laird, C.D., "An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition", to appear in Computers and Chemical Engineering, 2014.

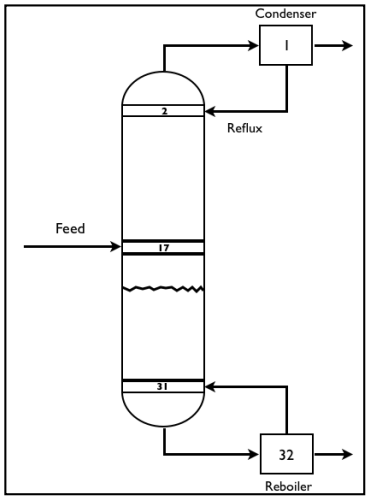
Dynamic Optimization
- block structure because of finite element discretization

$$\min_u \int_{t_0}^{t_f} L(x,y,u)\,dt$$
$$\text{s.t. } F(\dot{x},x,y,u) = 0$$
$$x(t_0) = x_0$$
$$(x,y,u)^L \leq (x,y,u) \leq (x,y,u)^U$$



Word, D.P., Kang, J., Akesson, J., and Laird, C.D., "Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems", to appear in Computational Optimization and Applications, 2014.

# Parallel Performance: Optimization Under Uncertainty



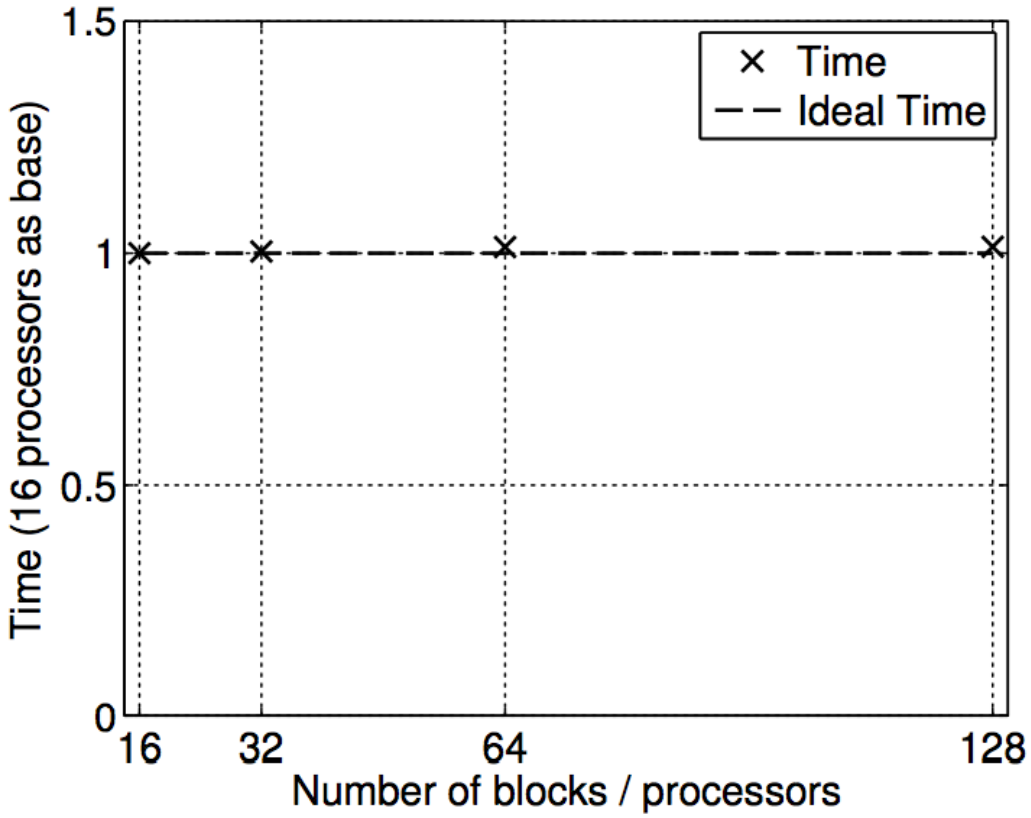

- 32 state variables, 35 algebraic variables
- Discretize model (OCFE)
- Uncertainty in mole fraction of the feed stream
- 96 scenarios, 32 processors

[Benallou, Seborg, and Mellichamp (1986)]

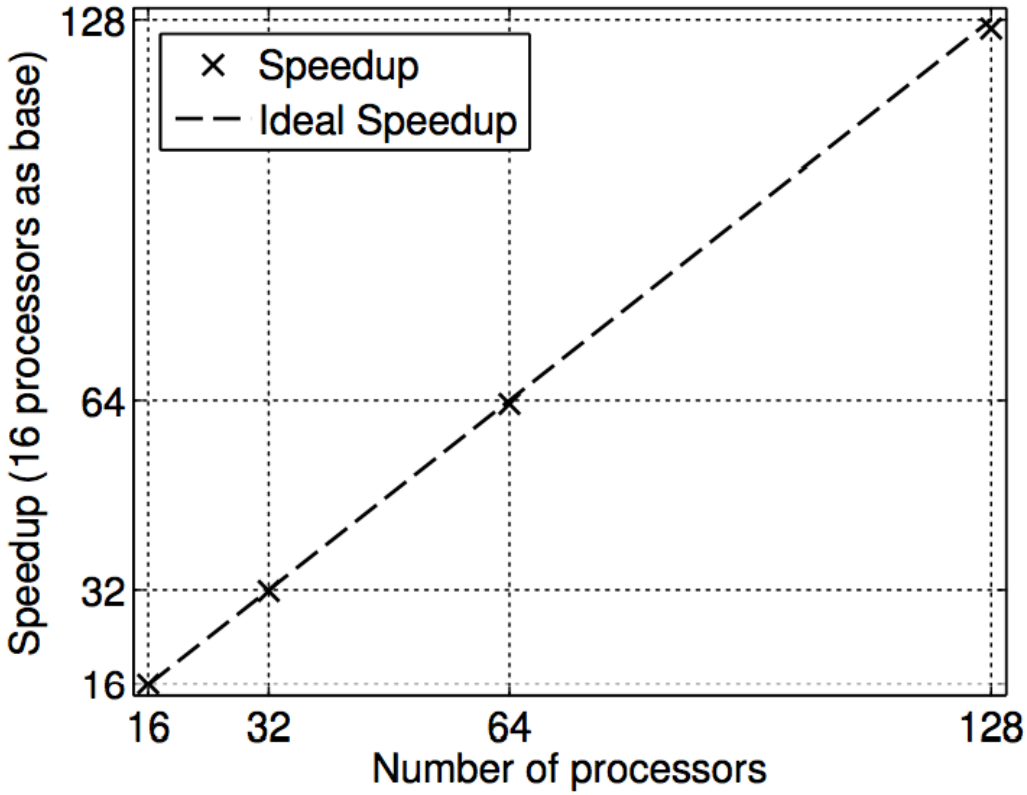
| Case | # Vars. | # Coupling Vars. | FS-S time(s) | ESC-S time(s) | ESC-P time(s) | PCGSC-S time(s) | PCGSC-P time(s) |
|---|---|---|---|---|---|---|---|
| 1 | 1430550 | 150 | 10.3 | 79.1 | 2.6 | 17.9 | 0.6 |
| 2 | 2861100 | 300 | - | - | 10.8 | - | 1.1 |
| 3 | 4291650 | 450 | - | - | 32.1 | - | 2.4 |
| 4 | 5722200 | 600 | - | - | 70.3 | - | 3.2 |
| 5 | 7152750 | 750 | - | - | 90.5 | - | 4.3 |
| 6 | 8583300 | 900 | - | - | 160.5 | - | 5.3 |
| 7 | 10013850 | 1050 | - | - | 218.0 | - | 6.3 |
| 8 | 11444400 | 1200 | - | - | 286.6 | - | 8.1 |

Kang, J., Word, D.P., and Laird, C.D., "An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition", to appear in Computers and Chemical Engineering, 2014.

# Weak and Strong Scaling: Optimization Under Uncertainty



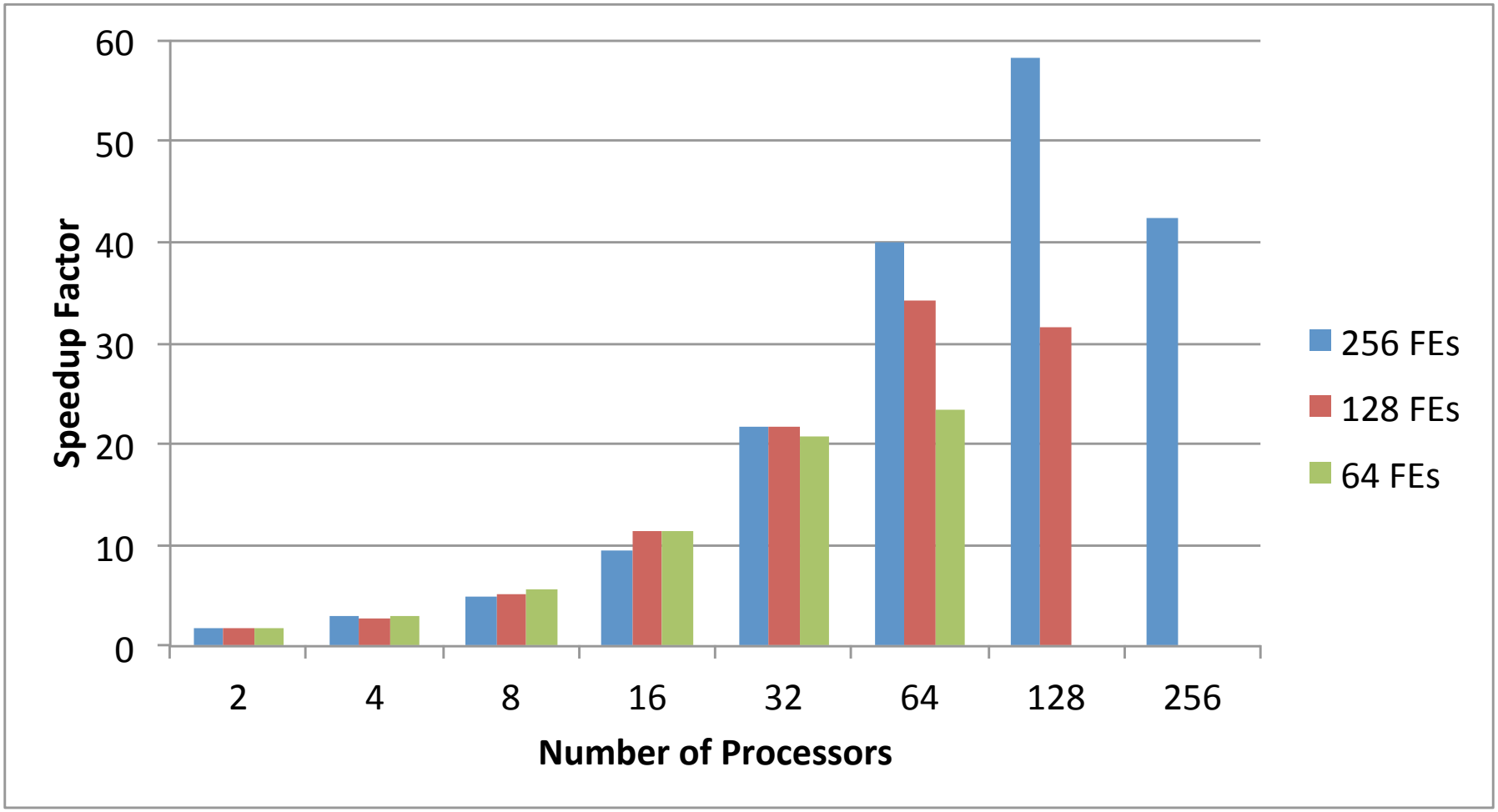

(a) Weak Scaling




(b) Strong Scaling

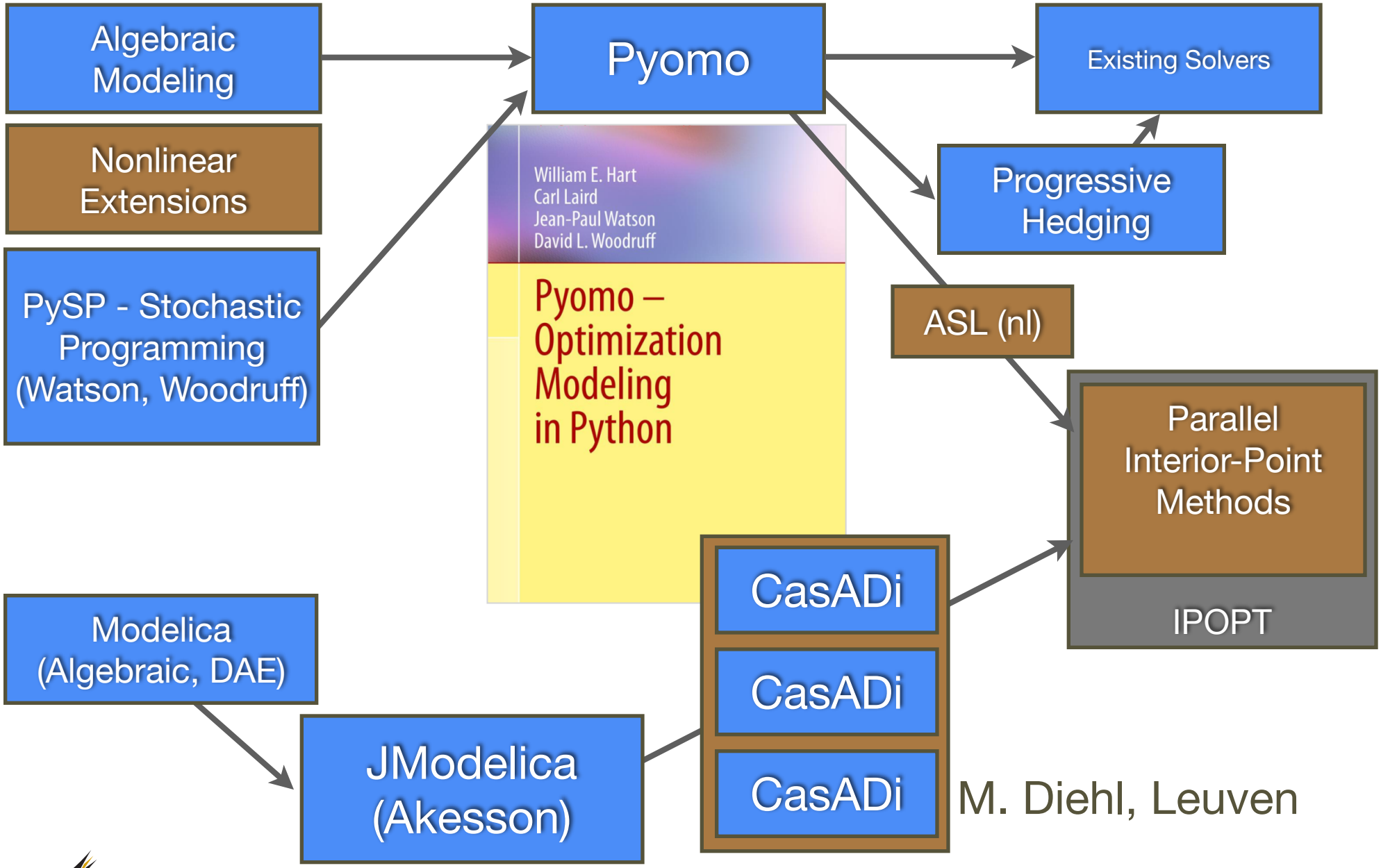
# Strong Scaling: Dynamic Optimization

Combined Cycle Power Plant with10 States and 127 Algebraic Constraints




Word, D.P., Kang, J., Akesson, J., and Laird, C.D., "Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems", to appear in Computational Optimization and Applications, 2014.

# Modeling Interfaces



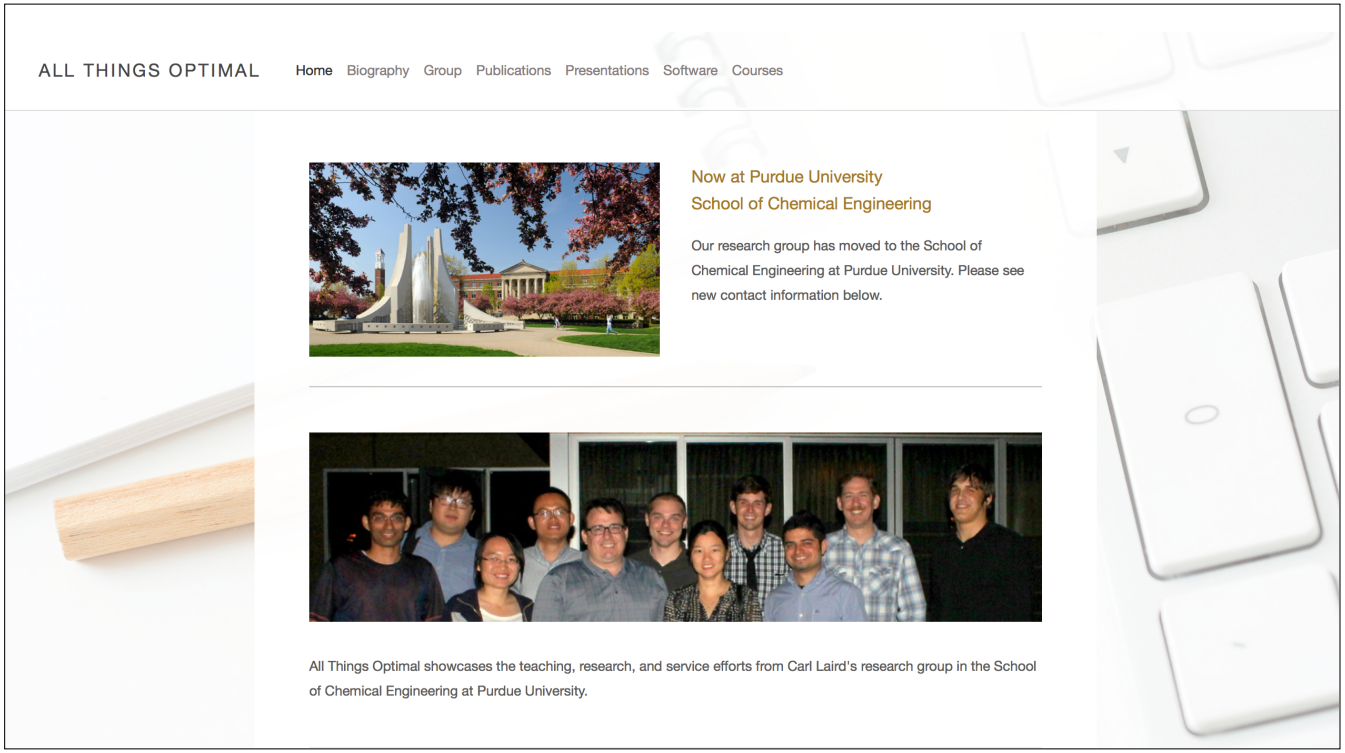
# Summary and Conclusions

- Demand for large-scale mathematical programming
  - Large-scale optimization advances allow new strategies
  - For continued improvement, parallel algorithms are necessary

- Applications, Architectures, Algorithms, Adoption
  - Need to understand the applications, assumptions
    - Uncertainty, discretization, spatial/network, data
  - Need to understand architectures
    - E.g. Big-Iron clusters, shared-memory multi-core, GPU
    - Not all parallel architectures created equal: strengths and limitations
  - Need for new algorithms
    - Tailored decomposition based on interior-point methods
    - Problem level decomposition strategies
    - Strategies are problem and architecture dependent
  - Need to make tools available for other researchers
    - Open-source parallel algorithms
    - Pyomo: Optimization Modeling in Python

# Acknowledgments

- Current Students/Researchers
  - **Jia Kang**
  - Arpan Seth
  - **Yankai Cao**
  - Alberto Benavides-Serrano
  - Jianfeng Liu
  - Michael Bynum
  - Todd Zhen
- Former Students/Researchers
  - **Yu Zhu**
  - Ahmed Rabie
  - George Abbott III
  - Chen Wang
  - Sean Legg
  - Daniel Word
  - Angelica Wong
  - Xiaorui Yu
  - **Gabriel Hackebeil**
  - Shawn McGee
- Collaborators
  - D. Cummings - JHSPH
  - S. Iamsirithaworn - MPHT
  - W. Hart, S. McKenna, J.P. Watson, K. Klise, John Siirola - Sandia
  - T. Haxton, R. Murray - EPA
  - Johan Akesson, Lund University
  - Sam Mannan, TAMU




Support
- National Science Foundation Cyber-Enabled Discovery and Innovation (CDI)-Type II
- National Science Foundation (CAREER Grant CBET# 0955205).
- Sandia National Laboratories, EPA, PUB Singapore
- MKOPSC, P2SAC