



**MOOCHO**  
and  
**Object-Oriented Interfaces and Algorithms for SAND PDE-  
Constrained Nonlinear Programming**

**Roscoe A. Bartlett**

**Senior Member Technical Staff  
Optimization and Uncertainty Estimation Department  
Sandia National Laboratories**



## Outline

---

- Overview of MOOCHO
- Requirements for levels of optimization => Linear algebra Interfaces
- Vector reduction/transformation operators (RTOp)
- Timing experiments



## Outline

---

- Overview of MOOCHO
- Requirements for levels of optimization => Linear algebra Interfaces
- Vector reduction/transformation operators (RTOp)
- Timing experiments



# Nonlinear Programming for Large-Scale Optimization

---

**Nonlinear program (NLP) formulation:**

$$\begin{array}{lll} \min & f(x) & f(x) : \mathbf{R}^n \rightarrow \mathbf{R} \\ \text{s.t.} & c(x) = 0 & c(x) : \mathbf{R}^n \rightarrow \mathbf{R}^m \\ & x_L \leq x \leq x_U & \end{array}$$

- **Classes of problems considered:**

- Large-scale (up to  $10^6$  variables and more)
- Special problem structure (e.g. PDEs, DAEs)
- Application specific methods (e.g. linear solvers, nonlinear globalization)
- Specialized computing environments (e.g. MPP, client/server, out-of-core)

- **What we can not do:**

- Use the current generation of optimization software to exploit problem structure and take advantage of advanced computing environments

- **What we do not want to do:**

- Write a new optimization implementation for each piece of application software

- **What we do want to do:**

- Use gradient-based methods for simultaneous analysis and design (SAND)



## Next Generation Optimization Software

---

### Desirable features:

- Provide sophisticated well-tested methods (NLP independent)
  - Globalization methods (i.e. line searches and/or trust regions)
  - Step modifications (i.e. negative curvature)
- Allow users to specialize parts of the algorithm that are NLP specific
  - Linear solvers (direct or iterative, serial or parallel)
  - Data structures (serial or parallel) => Fully scalable algorithms





## Outline

---

- Overview of MOOCHO
- Requirements for levels of optimization => Linear algebra Interfaces
- Vector reduction/transformation operators (RTOp)
- Timing experiments



# Requirements for Simultaneous Analysis and Design (SAND)

## Standard NLP formulation

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & c(x) = 0 \\ & x_L \leq x \leq x_U \end{array} \quad \begin{array}{l} f(x) : \mathcal{X} \rightarrow \mathbf{R} \\ c(x) : \mathcal{X} \rightarrow \mathcal{C} \\ \mathcal{X} \subseteq \mathbf{R}^n, \mathcal{C} \subseteq \mathbf{R}^m \end{array}$$

## NLP formulation for PDE-Constrained Optimization

$$\begin{array}{ll} \min & f(y,u) \\ \text{s.t.} & c(y,u) = 0 \\ & (y_L, u_L) \leq (y, u) \leq (y_U, u_U) \end{array} \quad \begin{array}{l} f(y,u) : \mathcal{Y} \times \mathcal{U} \rightarrow \mathbf{R} \\ c(y,u) : \mathcal{Y} \times \mathcal{U} \rightarrow \mathcal{C} \\ \mathcal{Y} \subseteq \mathbf{R}^m, \mathcal{U} \subseteq \mathbf{R}^{n-m}, \mathcal{C} \subseteq \mathbf{R}^m \end{array} \quad \text{where: } \mathcal{X} = \mathcal{Y} \times \mathcal{U}$$

## Definitions

- Basis  $C$  and nonbasis  $N$  matrices :  $\nabla c^T = [ C, N ]$  s.t.  $C$  is nonsingular
- Hessian of the Lagrangian  $W$  :  $W = \nabla^2 f(x) + \sum \lambda_j \nabla^2 c_j(x)$

## Requirements (Levels of Opt. Methods)

### Direct SAND

$$\begin{array}{ll} \text{Objective evaluation} & : (y_k, u_k) \rightarrow f_k \\ \text{Residual evaluation} & : (y_k, u_k) \rightarrow c_k \\ \text{Objective gradients} & : (y_k, u_k) \rightarrow \nabla f_k \\ \text{Newton step} & : p_k = -C_k^{-1} c_k \\ \text{Direct sensitivities} & : D_k = -C_k^{-1} N_k \end{array}$$

### Adjoint SAND

$$\begin{array}{ll} q = C p \quad \text{and} \quad p = C^T q, & \text{where: } p \in \mathcal{Y}, q \in \mathcal{C} \\ q = N p \quad \text{and} \quad p = N^T q, & \text{where: } p \in \mathcal{U}, q \in \mathcal{C} \\ q = C^{-1} p \quad \text{and} \quad q = C^{-T} p, & \text{where: } p \in \mathcal{C}, q \in \mathcal{Y} \end{array}$$

### Full-Newton SAND

$$\begin{array}{ll} q = W_{yy} p, & \text{where: } p \in \mathcal{Y}, q \in \mathcal{Y} \\ q = W_{yu} p \quad \text{and} \quad p = W_{yu}^T q, & \text{where: } p \in \mathcal{U}, q \in \mathcal{Y} \\ q = W_{uu} p, & \text{where: } p \in \mathcal{U}, q \in \mathcal{U} \end{array}$$

# Vectors!!!

# MOOCHO NLP (Application) Interfaces

Question: How are following represented?

Vectors spaces :  $\mathcal{X}, \mathcal{C}$   
 Vectors :  $x, x_L, x_U, c, \nabla f, p$   
 Matrices :  $\nabla c, N, D, W$   
 Invertible matrices :  $C$

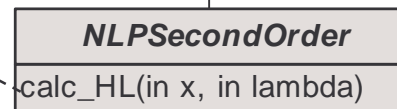
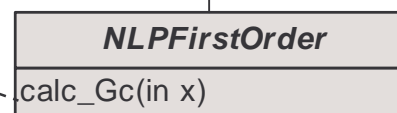
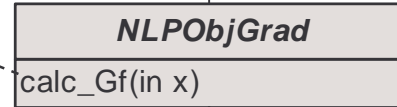
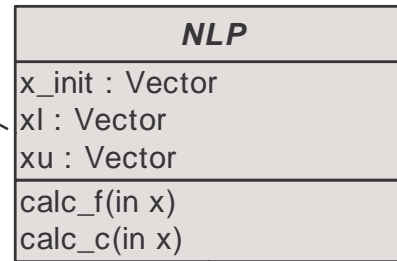
$\min f(x)$   
 s.t.  $c(x) = 0$   
 $x_L < x < x_U$

\* Initial guess  $x_0$   
 \* Variable bounds  $x_L, x_U$   
 \* Function evaluations  $f(x), c(x)$

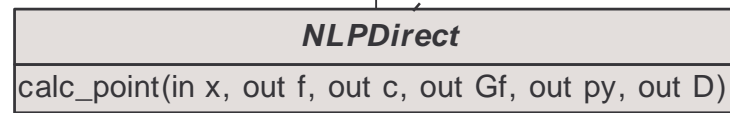
\* Objective gradient  $\nabla f(x)$

\* General constraints  
 gradients matrix  $\nabla c(x)$

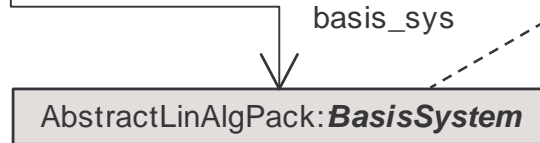
\* Hessian of the Larangian  
 matrix  
 $W = \nabla^2 f(x) + \sum \lambda_j \nabla^2 c_j(x)$



\* Direct  
 sensitivities  $D = -C^{-1} N$   
 \* Newton step  $p = -C^{-1} c$

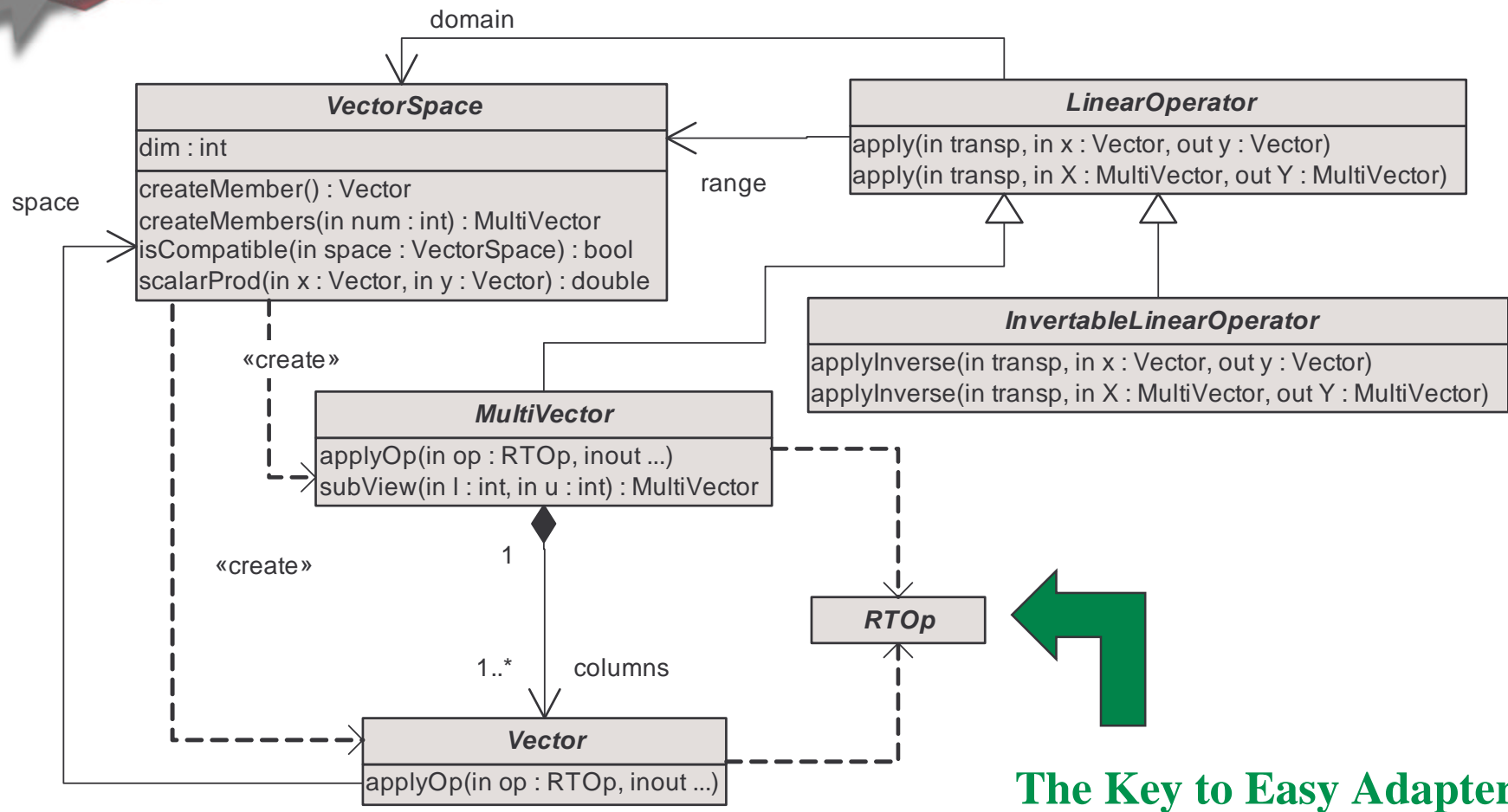


\* Basis  $C$  and nonbasis  $N$   
 handling  $\nabla c^T = [C \ N]$



Unified Modeling Language (UML) Class Diagram

# Linear Algebra Model for Application Interfaces



## (Nearly) Compatible Interface (C++) Implementations

- HCL (Hilbert Class Library)
- TSF (Trilinos Solver Framework)
- AbstractLinAlgPack => MOOCHO linear algebra interface



## Outline

---

- Overview of MOOCHO
- Requirements for levels of optimization => Linear algebra Interfaces
- **Vector reduction/transformation operators (RTOp)**
- Timing experiments



# Examples of Non-Standard Vector Operations

## Examples from OOQP (Gertz, Wright)

$$y_i \leftarrow y_i + \alpha x_i z_i, i = 1 \dots n$$

$$y_i \leftarrow y_i / x_i, i = 1 \dots n$$

$$y_i \leftarrow \begin{cases} y^{\min} - y_i & \text{if } y_i < y^{\min} \\ y^{\max} - y_i & \text{if } y_i > y^{\max} \\ 0 & \text{if } y^{\min} \leq y_i \leq y^{\max} \end{cases}, i = 1 \dots n$$

$$\alpha \leftarrow \{ \max \alpha : x + \alpha d \geq \beta \}$$

## Example from TRICE (Dennis, Heinkenschloss, Vicente)

$$d_i \leftarrow \begin{cases} (b - u)_i^{1/2} & \text{if } w_i < 0 \text{ and } b_i < +\infty \\ 1 & \text{if } w_i < 0 \text{ and } b_i = +\infty \\ (u - a)_i^{1/2} & \text{if } w_i \geq 0 \text{ and } a_i > -\infty \\ 1 & \text{if } w_i \geq 0 \text{ and } a_i = -\infty \end{cases}, i = 1 \dots n$$

## Example from IPOPT (Wächter)

$$x_i \leftarrow \begin{cases} \left( x_i^L + \frac{(x_i^U - x_i^L)}{2} \right) & \text{if } \hat{x}^L_i > \hat{x}^U_i \\ \hat{x}^L_i & \text{if } x_i < \hat{x}^L_i \\ \hat{x}^U_i & \text{if } x_i > \hat{x}^U_i \end{cases}, i = 1 \dots n$$

Currently in MOOCHO :  
> 40 vector operations!

where :

$$\hat{x}^L_i = \min \left( x_i^L + \eta (x_i^U - x_i^L), x_i^L + \delta \right)$$
$$\hat{x}^U_i = \max \left( x_i^L - \eta (x_i^U - x_i^L), x_i^U - \delta \right)$$



## Goals for a Vector Interface

---

Compute efficiency

=> Near optimal performance

Optimization developers add new operations

=> Independence of linear algebra library developers

Compute environment independence

=> Flexible optimization software

Minimal number of methods

=> Easy to write adapters



## Approaches to Developing Vector Interfaces

---

- (1) Linear algebra library allows direct access to vector elements
  
- (2) Optimizer-specific interfaces
  
- (3) General-purpose primitive vector operations



## Vector Reduction/Transformation Operators Defined

### Reduction/Transformation Operators (RTOp) Defined

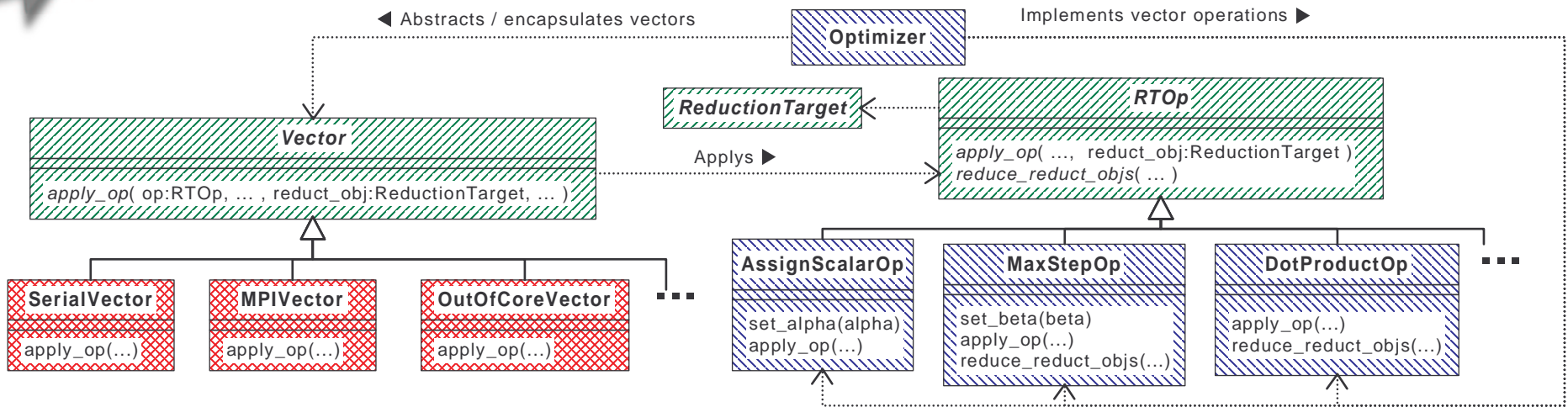
$z^1_i \dots z^q_i$	$\leftarrow \text{op}_t(i, v^1_i \dots v^p_i, z^1_i \dots z^q_i)$	element-wise transformation
$\beta$	$\leftarrow \text{op}_r(i, v^1_i \dots v^p_i, z^1_i \dots z^q_i)$	element-wise reduction
$\beta^2$	$\leftarrow \text{op}_{rr}(\beta^1, \beta^2)$	reduction of intermediate reduction objects

- $v^1 \dots v^p \in \mathbf{R}^n$  :  $p$  non-mutable input vectors
- $z^1 \dots z^q \in \mathbf{R}^n$  :  $q$  mutable input/output vectors
- $\beta$  : reduction target object (many be non-scalar (e.g.  $\{y_k, k\}$ ), or **NULL**)

### Key to Optimal Performance

- $\text{op}_t(\dots)$  and  $\text{op}_r(\dots)$  applied to entire sets of subvectors ( $i = a \dots b$ ) independently:  
 $z^1_{a:b} \dots z^q_{a:b}, \beta \leftarrow \text{op}(a, b, v^1_{a:b} \dots v^p_{a:b}, z^1_{a:b} \dots z^q_{a:b}, \beta)$
- Communication between sets of subvectors only for  $\beta \neq \text{NULL}$ ,  $\text{op}_{rr}(\beta^1, \beta^2) \rightarrow \beta^2$

# Object-Oriented Design for User Defined RTOp Operators



## Advantages:

- Functionality
  - **Linear-algebra implementations** can be changed with no impact on **optimizer**
  - **Optimizer developers** can unilaterally add new vector operations
- Performance
  - Near optimal performance (large subvectors)
  - Multiple simultaneous global reductions => no sequential bottlenecks
  - No unnecessary temporary vectors or multiple vector read/writes
- **Disadvantages:**
  - New concepts, initially harder to understand interfaces?



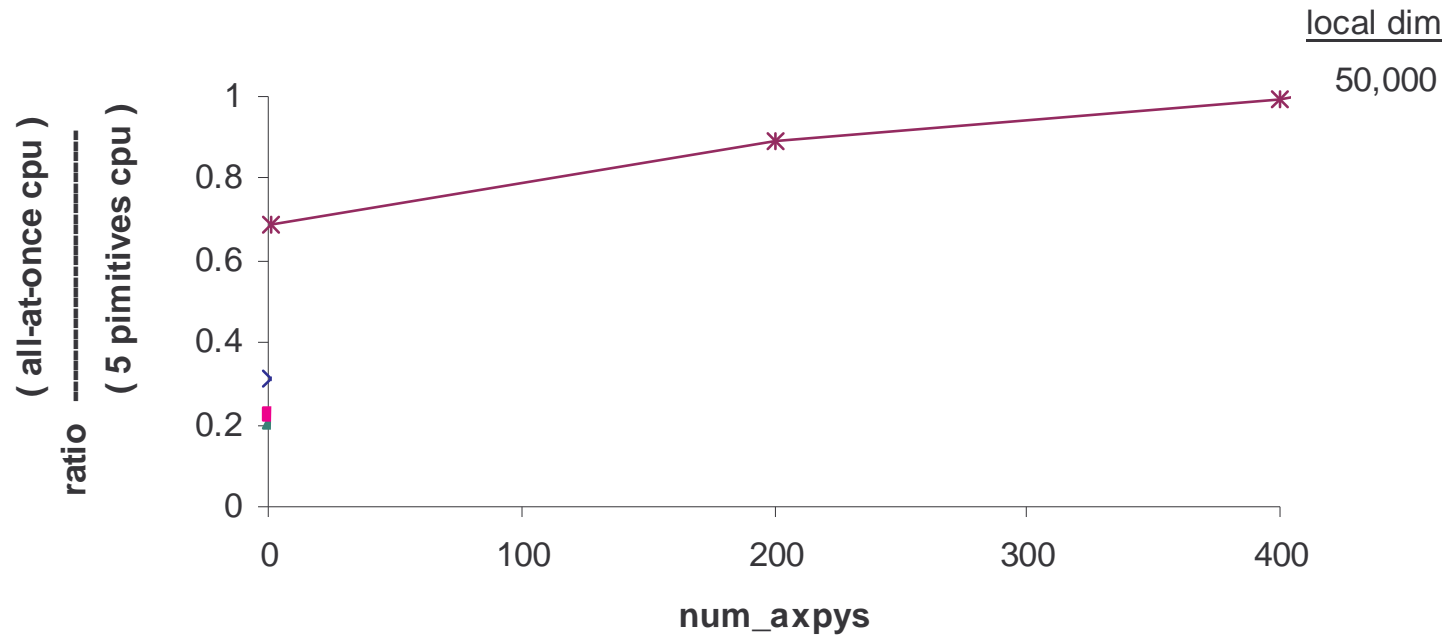
## Outline

---

- Overview of MOOCHO
- Requirements for levels of optimization => Linear algebra Interfaces
- Vector reduction/transformation operators (RTOp)
- **Timing experiments**



## RTOp vs. Primitives : Communication



\* 128 processors on CPlant<sup>®</sup>

- **Compare**

- **RTOp (all-at-once reduction (i.e. ISIS++ QMR solver))**

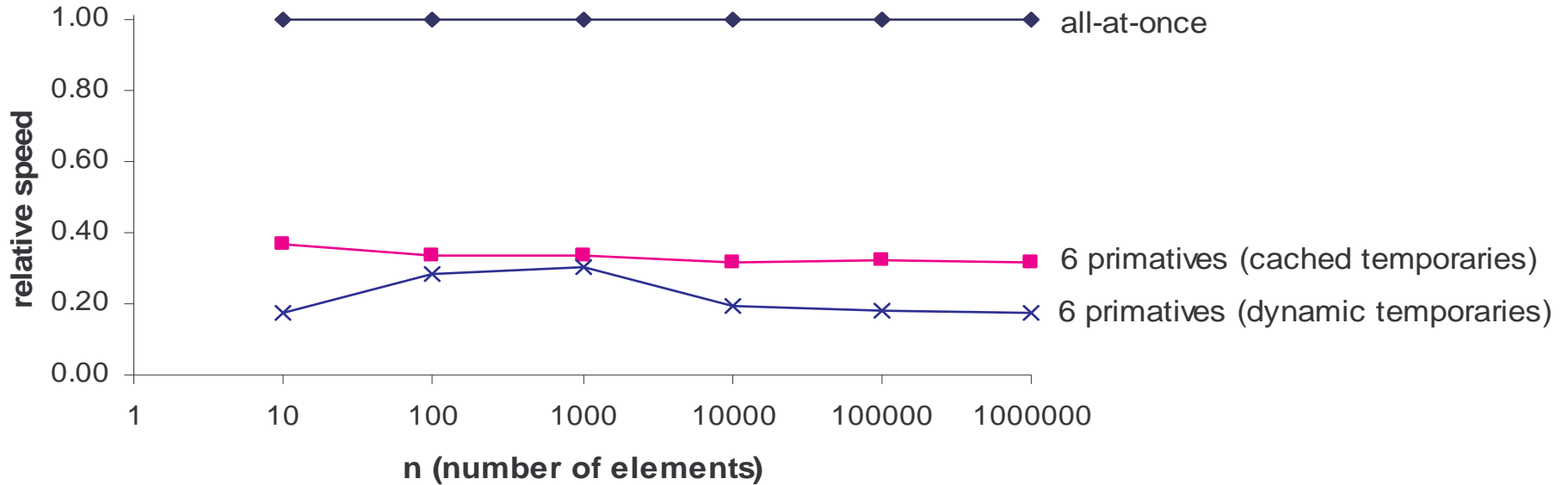
$$\{ \alpha, \gamma, \xi, \rho, \varepsilon \} \leftarrow \{ (x^T x)^{1/2}, (v^T v)^{1/2}, (w^T w)^{1/2}, w^T v, v^T t \}$$

- **Primitives (5 separate reductions)**

$$\alpha \leftarrow (x^T x)^{1/2}, \gamma \leftarrow (v^T v)^{1/2}, \xi \leftarrow (w^T w)^{1/2}, \rho \leftarrow w^T v, \varepsilon \leftarrow v^T t$$



## RTOp vs. Primitives : Multiple Ops and Temporaries



\* 1 processor (gcc 3.1 under Linux)

- **Compare**

- **RTOp (all-at-once reduction)**

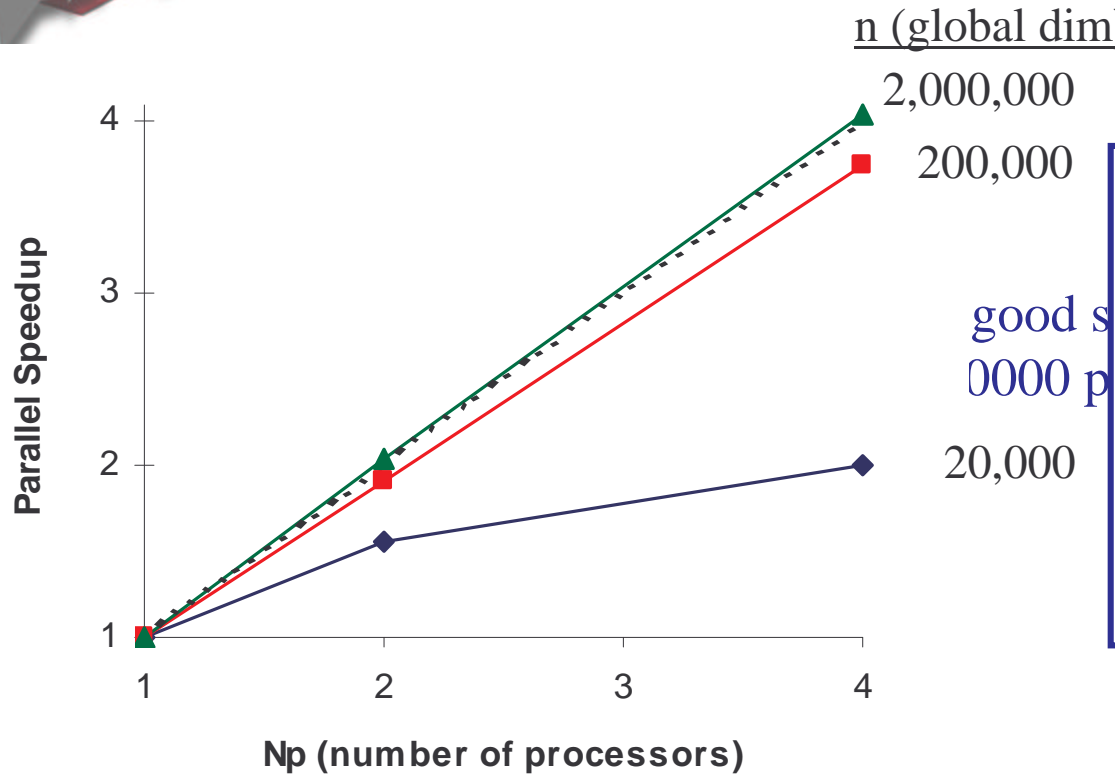
$$\{ \max \alpha : x + \alpha d \geq \beta \} = \min \{ \max( (\beta - x_i) / d_i, 0 ), \text{ for } i = 1 \dots n \} \rightarrow \alpha$$

- **Primitives (5 temporaries, 6 vector operations)**

$$-x_i \rightarrow u_i, \quad x_i + \beta \rightarrow v_i, \quad v_i / d_i \rightarrow w_i, \quad 0 \rightarrow y_i, \quad \max\{w_i, y_i\} \rightarrow z_i, \quad \min\{z_i, i=1 \dots n\} \rightarrow \alpha$$



# Parallel Scalability of MOOCHO



**Where is the parallel bottleneck?**

**Is it OO C++ or MPI? Answer => MPI**

**Serial overhead of MOOCHO (n=2, N<sub>p</sub>=1)**  
 ≈ 0.41 milliseconds per rSQP iteration

**Overhead of MPI communication (N<sub>p</sub>=4)**  
 ≈ 0.42 milliseconds per global reduction

- \* Red Hat Linux cluster (4 nodes)**
- 2.0 GHz Intel P4 processors
  - MPICH 1.2.2.1

Scaleable example  
NLP (m = n/2)

$$\min f(x) = \frac{1}{2} \sum_{i=1}^n x_i^2$$

$$s.t. \quad c_j(x) = x_j(x_{m+j} - 1) - 10x_{m+j} = 0 \quad j = 1, \dots, n/2$$

Variable reduction  
range / null space  
decomposition

$$A^T = [C \quad N] = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

$$Z = \begin{bmatrix} -C^{-1}N \\ I \end{bmatrix}$$

- Diagonal matrices => All vector ops!



## Summary & Conclusions

---

### MOOCHO

- MOOCHO is framework/library for large-scale NLP
- MOOCHO currently supports several active-set and interior-point SQP-related methods
- MOOCHO can be adapted to the application
- MOOCHO is fully scalable

### Interoperability of optimization applications and algorithms

- Linear algebra interfaces are key to interoperability between applications and numerical algorithms

Needed: Common application **linear algebra model**

Needed: Common interface for vector operations (**RTOp**)



## Availability & Future Work

---

### Websites

- MOOCHO

- <http://cs.sandia.gov/MOOCHO> (spring/summer 2003)

- RTOp

- <http://cs.sandia.gov/MOOCHO/RTOp> (spring 2003)

### Future Work

- Optimization algorithms/software

- Full-space direct (Biegler & Laird)
  - Full-space iterative (help???)
  - Interior point (Biegler, Laird & Waechter)

- Applications

- Sundance (ongoing)
  - Shape optimization (Euler equations with Edge/Reshape)
  - Shape optimization (Euler equations with Premo)
  - Circuit simulation (Xyce)



**The End**

---

**Thank You!**