

## Optimization on Computational Grid

STEVE WRIGHT

Math and Computer Science Division / Argonne National  
Computer Science Department / University of Chicago  
wright@mcs.anl.gov

<http://www.mcs.anl.gov/metaneos>

## Brains of The Outfit



{ JEAN-PIERRE GOU  
JEFF LINDEROTH

{ QUN CHEN  
MICHAEL FERRIS  
SANJEEV KULKARNI  
MIRON LIVNY  
MIKE YODER

## More Brains of The Outfit



{ MARCEL GOOD  
JEAN-PIERRE GOU  
KURT ANSTREICHE  
NATE BRIXIUS

## Topics

- Goals and Motivation for metaNEOS
- Target problem classes: Matching the platform with the algorithm and problem class
- MW
- Case I: Integer LP
- Case II: QAP
- Case III: Stochastic LP
- iMW
- Conclusions and Plans

## Computational Grid (a.k.a. Metacompu

A computational grid is a collection of computers (and mu visualization and storage devices) that are geographically distributed, but networked in various ways.

- Dynamic availability of processors
- Unreliability
- Poor communications properties
- Heterogeneity
- Scale.

We focus on *inexpensive* platforms, especially the Condor which delivers a parallel computing environment consistin time on networks of workstations.

## Goals of metaNEOS

1. Design metacomputing environments and tools suitable solving large optimization problems,
2. Identify optimization problems that fit these environments, develop algorithms to solve them, and
3. Use the resulting tools to solve problems of unprecedented and complexity.

## 2. Target Problem Classes and Algorithms

Tackle only problem classes and algorithms for which the metacomputing environment is really useful:

- Very large scale
- Compute-intensive rather than data-intensive
- Computation is divisible into a large number of tasks require little synchronicity.

- **Global Optimization:** multistart smoothing algorithm (Moré, Zakeri);
- **Integer Linear Programming:** branch-and-bound implemented as master-worker (Chen, Ferris, Linderoth);
- **Stochastic Programming:** verification (Zakeri);
- **Integer Nonlinear Programming:** branch-and-bound implemented as master-worker (Goux, Leyffer, Nocedal);
- **Stochastic Programming:** L-shaped decomposition algorithm implemented as master-worker (Linderoth, Linderoth);
- **Quadratic Assignment Problem:** branch-and-bound QP bounding and specialized branching, implemented as master-worker (Anstreicher, Brixius, Goux, Linderoth);

## 1. Platforms and Tools

metaNEOS includes researchers from two metacomputing

- Globus: [www.globus.org](http://www.globus.org)
- Condor: [www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor)

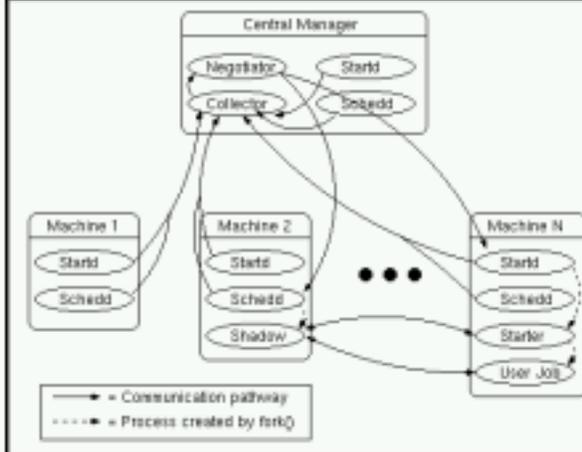
Globus project is developing basic software infrastructure (middleware) for computations that integrate geographically distributed computational and information resources.

Condor provides software tools that support high through computing on distributively owned resources (w.g. workst a campus). Can use Globus to grab resources for a user's (Personal Condor).

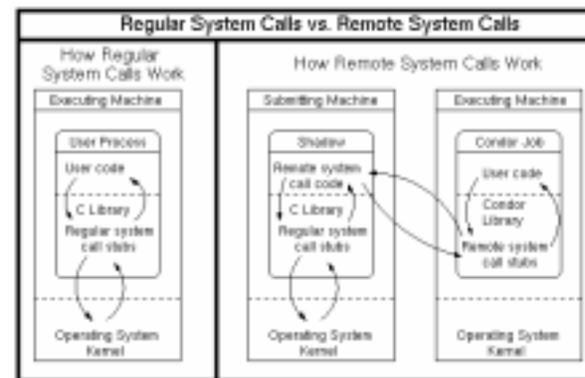
## Standard Condor

- User submits job from submitting machine (S); Condor Manager (C) finds an executing machine (E) in the C pool and activates a "starter" on this machine.
- Starter obtains binary from S and runs it on E.
- Condor traps system calls (e.g. I/O) and processes them through S instead of E; E's file system is protected.
- Starter checkpoints the job periodically (sends checkpoints back to S or to a checkpoint server).
- If job is killed on E (e.g. user of E starts to run some other job), Condor finds a new E and restarts job from the last checkpoint (*migration*).

Architecture of a Condor Pool  
(With a job submitted on Machine 2 running on Machine N)



## Remote System Calls



## Parallel Computation using Condor

Machine  $S$  can send a bunch of jobs to the Condor pool and execute on  $E_1, E_2, \dots, E_N$ . Opens the door to parallel algorithms.

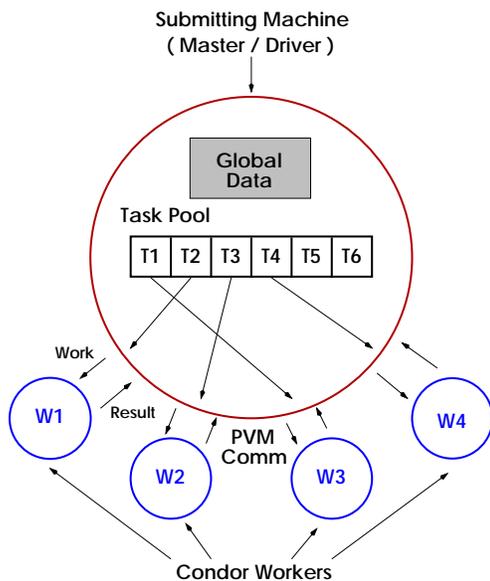
Since communication can occur only as  $S \leftrightarrow E_1, S \leftrightarrow E_2$ , paradigms that require lots of interprocessor communication are difficult to support.

However, paradigms such as task-farming and master-worker (master running on  $S$ ) can be supported easily—in principle.

*To use Condor effectively, need an API suitable for implementation of parallel optimization algorithms that use the Master-Worker paradigm.*

## MW: Master-Worker

- MW is an API that supports Master-Worker computation on Condor platforms.
- Uses Condor/PVM (PVM=Parallel Virtual Machine) and adds three functions to PVM:
  - ◊ HostAdd
  - ◊ HostSuspend
  - ◊ HostDelete
- User is responsible for taking appropriate actions in response to these things happening.
- Developers: Yoder, Linderoth, Goux.



## MW Abstraction

- MWMaster
  - ◊ `get_userinfo()`
  - ◊ `setup_initial_tasks()`
  - ◊ `pack_worker_init_data()`
  - ◊ `act_on_completed_task()`
- MWTask
  - ◊ `(un)pack_work`
  - ◊ `(un)pack_result`
- MWWorker
  - ◊ `unpack_worker_init_data()`
  - ◊ `execute_task()`

## MWApplications

- MWFATCOP – A branch and cut code for linear integer programming
- MWMINLP – A branch and bound code for nonlinear programming
- MWSVM – A column generation code for solving support vector machine problems
- MWLShaped – A cutting plane and verification code for stochastic programming
- MWQAP – A branch and bound code for solving the assignment problem
- ... (Your application here) ...

## FATCOP

- Originally used Condor/PVM directly; now uses MW
- Implements the various steps associated with a branch and bound approach: preprocessing, branching, bounding, adding
- At any given time, we have a pool of tasks (each task is a relaxation) and workers (each worker is a machine that acquires a task through Condor/PVM and that processes it until it goes away).
- The LP code used by each worker does not have to be the same. (SOPLEX, CPLEX).
- Developers: Chen and Ferris (Wisconsin) + Linderot (University of version).

Instance	Statistic	$\bar{E}$	$\bar{P}$	Nodes	Time
pk1	minimum	63.5	6.7	2786670	1275
	average	72.3	13.4	3511512	2138
	maximum	78.9	19.0	4048204	4353
ll52lav	minimum	82.4	1.9	1845	53
	average	91.6	6.5	2824	572
	maximum	98.1	16.9	3919	2080
stein45	minimum	45.6	2.9	181986	499
	average	64.1	12.3	194472	1018
	maximum	88.7	25.0	215029	1694
p2756	minimum	44.3	7.6	6249	866
	average	51.3	21.1	9687	1595
	maximum	61.9	14.4	14115	2202
air05	minimum	18.9	28	7779	3503
	average	43.6	42.4	8180	5771
	maximum	69.3	58.6	8419	9549
bell5	minimum	33.3	6.6	518875	359
	average	42.0	11.8	1600810	1098
	maximum	54.2	14.4	457876	2464
gess2.o	minimum	72.7	43.2	6207993	6951
	average	84.9	62.5	8214031	10074
	maximum	94.6	86.3	9693518	13198

Table 5: Performance of FATCOP 2.0

## Stochastic Programming

(Linderoth). Two-Stage Linear Recourse Program

$$\begin{aligned} \min \quad & c^T x + \sum_{k=1}^K p_k q_k^T y_k, \\ \text{s.t.} \quad & Ax = b, \\ & T_k x + W y_k = h_k, \quad k = 1, 2, \dots, K, \\ & x \geq 0, \quad y_k \geq 0, \quad k = 1, 2, \dots, K. \end{aligned}$$

- $K$ : A number of “scenarios”
- $x$ : First-stage variables
- $y_k$ : Second-stage variables
- $p_k$ : Scenario probabilities

## Reformulating

- Can also express as a problem with a piecewise-linear  $Q(x)$  in the variables  $x$  only:

$$\min_x c^T x + Q(x), \quad \text{s.t. } Ax = b, x \geq 0,$$

where  $Q(x)$  is defined as

$$\begin{aligned} Q(x) &= \min_{y_1, y_2, \dots, y_K} \sum_{k=1}^K p_k q_k^T y_k, \\ \text{s.t.} \quad & W y_k = h_k - T_k x, \quad k = 1, 2, \dots, K, \\ & y_k \geq 0, \quad k = 1, 2, \dots, K. \end{aligned}$$

- Important Facts:
  - ◊  $Q(x)$  is a piecewise linear convex function of  $x$ .
  - ◊ Evaluation of  $Q(x)$  is a separable problem; can solve  $k = 1, 2, \dots, K$  independently.

## L-shaped (Benders) decomposition

- Represent  $Q$  by an artificial variable  $\theta$  and find supporting planes for  $\theta$ :  $\theta \geq \beta_\ell - \alpha_\ell^T x$ ,  $\ell = 1, 2, \dots, L$ .
  - ◊ Inequalities obtained as a byproduct of evaluating

### L-Shaped Method

1. Solve a **master problem** with the current  $\theta$ -approximation  $Q(x)$ , to obtain candidate first-stage solution  $\hat{x}$ ;
2. Fix  $x = \hat{x}$  and Solve the **subproblems** for  $y_1, y_2, \dots$ , (evaluating  $Q(\hat{x})$ ). Use subproblem solution data to add supporting planes for  $\theta$  approximation at  $\hat{x}$ .
3. Goto 1.

## Quasi-Multi-Cuts

- “Classical” L-Shaped  $\Rightarrow Q \sim \theta$

$$\theta \geq \sum_{k=1}^K p_k \pi_{k\nu}^T h_k - \sum_{k=1}^K p_k \pi_{k\nu}^T T_k x$$

- “Multicut”  $\Rightarrow Q \sim \sum_{k=1}^K \theta_k$

$$\theta_k \geq \pi_{k\nu}^T h_k - \pi_{k\nu}^T T_k x$$

- We can actually partition the scenarios any way we like  
“Quasi”-Multicut  $\Rightarrow Q \sim \sum_{j=0}^n \theta_{[\kappa_j \dots \kappa_{j+1}]}$ .

$$\theta_{[\kappa_j \dots \kappa_{j+1}]} \geq \sum_{k=\kappa_j}^{\kappa_{j+1}} p_k \pi_{k\nu}^T h_k - \sum_{k=\kappa_j}^{\kappa_{j+1}} p_k \pi_{k\nu}^T T_k x$$

## MWLShaped – LShapedMaster

- `get_userinfo()`
  - ◊ Reads initial data, solves the expected value problem, and provides a good guess for  $x$ .
- `pack_worker_init_data()`
  - ◊ Initial data are  $W$  and “compact” forms of  $p_k, q_k, \dots$
- `setup_initial_tasks()`
  - ◊ Solves first stage I LP.
  - ◊ Creates a batch of tasks for the first iteration.
- `act_on_completed_task()`
  - ◊ If all tasks for this iteration are complete, add the solution to solve the linear program and add a new batch of tasks for the next iteration.

## MWLshaped – LShapedTask

- Work
  - ◊ Starting scenario index ( $k_b$ ), and ending scenario index ( $k_e$ ).
  - ◊ Stage I solution ( $\hat{x}$ ).
  - ◊ Estimated expected stage II objective function value ( $\hat{\theta}_{[k_b, k_e]}$ ).
- Result
  - ◊ A cut, if the estimate of  $(\theta_{[k_b, k_e]})$  is not good enough.
- We form one cut per cluster of scenarios.
- ★ The computing environment has influenced our choice of algorithm!!

## MWLShaped – LShapedWorker

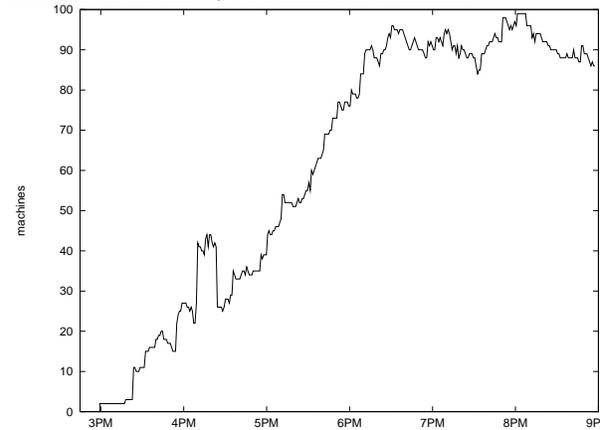
- `unpack_init_data()`
  - ◊ Unpacks the problems initial data and loads the “compact” stage II LP to the solver.
- `execute_task()`
  - ◊ Solve LPs for all scenarios  $k_b \leq k \leq k_e$ , and create a cut if possible.

## Avoiding Synchronicity

- Different processors act at different speeds – many wait for the “slowpoke”
  - ◊ “Balance”  $|k_e - k_b|$  with the speed of the machines
    - Involves adding both columns and rows to the model as machines arrive
- All workers wait for the master LP
  - ◊ Don’t have the master wait for all tasks!!
    - + If proportion  $\alpha$  of the tasks for an iteration have completed with cuts, start a new iteration
- ★ Again, the computing environment has influenced our algorithmic choices!!

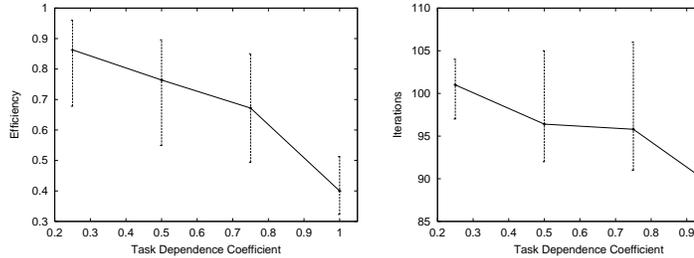
## Machines used in Stochastic Programming

Solved 20,000 scenario instance, whose “deterministic equivalent” has > 3 million rows and > 14 million columns



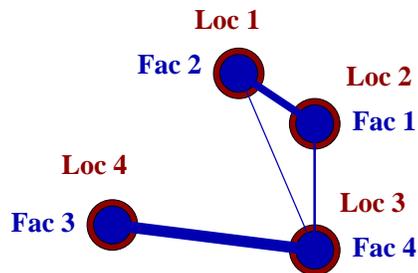
## $\alpha$ vs Efficiency and Iteration Count

- MWLShaped – Synchronicity  $\alpha$  vs efficiency and iteration count



## MWQAP

(Anstreicher, Brixius, Goux, Linderoth)



## QAP formulation

$$\min_{\pi \in \Pi} \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{\pi(i)\pi(j)} + \sum_{i=1}^n c_i \pi(i),$$

where

- $(\pi(1), \pi(2), \dots, \pi(n))$  is a permutation of  $(1, 2, \dots, n)$ ; the location to which facility  $i$  is assigned.
- $b_{kl}$  is the distance between locations  $k$  and  $l$ ;
- $a_{ij}$  is the flow between facilities  $i$  and  $j$ ;
- $c_{ik}$  is the cost of building facility  $i$  at location  $k$ .

$\mathcal{NP}$  hard problem;  $n!$  possibilities.

Example: NUG25: If  $10^{10}$  possible combinations evaluated per second, would still take 50 million years.

## Basic Strategy: Branch and Bound

- Branch by
  - fixing a facility  $i$ , and forming child nodes in which assigned to each available location in turn;
  - fixing location  $k$ , and forming child nodes in which available facility is assigned to  $k$ ;
  - either one (50% better)
- Lower bound by solving a continuous relaxation
  - Gilmore-Lawler bound
  - Quadratic Programming bound (Anstreicher/Bixi use it!)
  - Semidefinite Programming bound

## Enhancements

Branching Strategy:

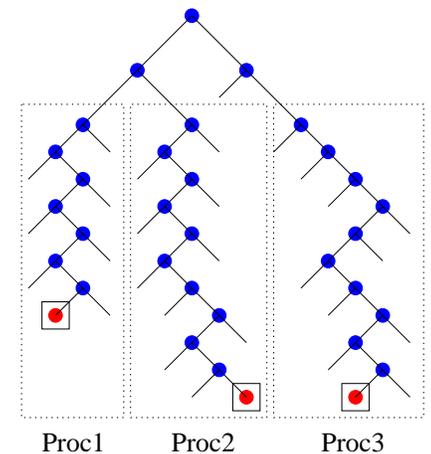
- Use dual information from QP, look-ahead, various heuristics
- Branching rules evolve with depth in tree, duality gap

Knuth's Estimator:

- Estimates size of B&B tree, CPU time;
- Helps decide whether problem solvable or not
- Helps tune algorithm parameters (branching/bounding)

## Parallel Strategy

- A single task consists of processing a subtree depth-first. Returns a task pool as result (or the solution), along with bound and statistical/performance information.
- Finish-Up Strategy: Take nodes that are deep in the tree (depth > 6) and traverse the subtree rooted at these nodes. (Usually fast tasks.) Reduces the number of tasks that are returned by this task.
- Child Reordering based on bound information.



## QAPLIB Challenge

[www.imm.dtu.dk/~sk/qaplib/](http://www.imm.dtu.dk/~sk/qaplib/)

NUG challenge problems, solved by Hahn et al. (serial machines) and Marzetta et al. (parallel machines).

NUG	Machine	Time	Solver
NUG22	Ultra 360Mhz	56 Hours	Hahn
NUG24	Ultra 360Mhz	9 days	Hahn
NUG25	Ultra 360Mhz	66 days	Hahn
NUG22	48-96 Cenju-3	9 days	Marzetta(1)
NUG25	64-128 Paragon	30 days	Marzetta(1)

## MWQAP Results: NUG25

NUG25: 6.7 hours! (January 20, 2000)

Machines	Linux / Solaris
Average # Machines	93
Equivalent Pool Performance	42
Parallel Efficiency	97%
Nodes	80,430,341
Total time	27 days
Normalized time	12.7 days

## MWQAP Results: NUG27

World Record Problem in the NUG class. Wall clock: 24 (February 2000).

Machines	Linux / Solaris
Average # Machines	136
Max # Machines	238
Equivalent Pool Performance	126
Parallel Efficiency	N/A
Nodes	513,160,139
Normalized Cumulated time	127 days

## World Records!

- Mixed Integer Nonlinear Programming
  - ◊ Solved previously unsolved instances of a trim-loss
- Stochastic Programming
  - ◊ Solved 20,000 scenario instance, whose “deterministic equivalent” has > 3 million rows and > 14 million
- Quadratic Assignment Problem
  - ◊ Solved a difficult (size 25) instance in a fraction of (> 3 months  $\Rightarrow$  < 7 hours)
  - ◊ Recently solved the world’s largest QAP! (size 27)
  - ◊ Gunning for the famous “nug30” instance – unsolved over 30 years.

## iMW

(Good and Goux.)

- Web-based problem solving environment
- Allows remote access to jobs running in MW:
  - ◊ Remote Submission
  - ◊ Remote Monitoring
  - ◊ Remote Steering
- Customizable interface

## Remote Submission

Web submission to server—like NEOS Server.

- Submits a form to iMW Apache server
- Java Servlet parses informatio
- Chooses unique ID, creates problem directory
- Compiles and submits problem to remote metacompu

## Remote Monitoring

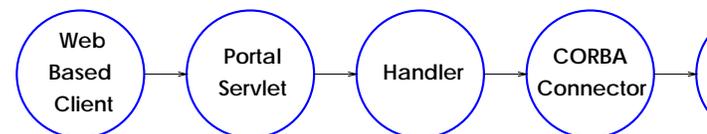
- User needs asynchronous feedback
- Master object contains control & status information
- Monitoring tool needs to communicate with master object  
needs to know its location.
- Seamless integration with object-oriented design of MW  
desirable!

### CORBA!

Design a multithreaded object that runs the Master object  
MW application, returns a reference to the Master object  
incoming CORBA calls.

## Web Interface

- Via generic URLs, user can submit, monitor, steer
- Java Servlets running on an Apache server provide po  
the iMW environment (The CORBA support available  
helps a lot..)
- How to transmit and format the monitoring informati  
the browser?



## HTML / Dynamic HTML

- ▶ Machine : **tarzan.cs.wisc.edu**
- ▶ Machine : **jane.cs.wisc.edu**
- ▶ Machine : **cheetah.cs.wisc.edu**

- State : SUSPENDED
- Operating system : SOLARIS26
- Architecture : INTEL
- RAM : 64
- Virtual Memory : 185724
- DiskSpace : 29794
- KFlops : 47464
- Mips : 221
- CPUs : 1
- Total time : 1356
- Total working time : 1324
- Total suspended time : 32
- Tasks handled : 54

```
<font face="Verdana,Helvetica,Arial" size=+1 color="#003399">
cheetah.cs.wisc.edu</b>
</font>
<font face="Verdana,Helvetica,Arial" color="#003399">
<ul>
<li>State : SUSPENDED</li>
<li>Operating system : SOLARIS26</li>
<li>Architecture : INTEL</li>
<li>RAM : 64 </li>
<li>Virtual Memory : 185724 </li>
<li>DiskSpace : 29794</li>
<li>KFlops : 47464</li>
<li>Mips : 221</li>
<li>CPUs : 1</li>
<li>Total time : 1356</li>
</ul>
</font>
```

Separate data description and display ~> XML

```
<Worker>
<WorkerPhysicalProperties>
  <WorkerName>cheetah.cs.wisc.edu</WorkerName>
  <WorkerStatus>SUSPENDED</WorkerStatus>
  <WorkerOpSys>"SOLARIS26"</WorkerOpSys>
  <WorkerArch>"INTEL"</WorkerArch>
  <WorkerMemory>64</WorkerMemory>
  <WorkerVirtualMemory>185724</WorkerVirtualMem
  <WorkerDiskSpace>29794</WorkerDiskSpace>
  <WorkerKFlops>47464</WorkerKFlops>
  <WorkerMips>221</WorkerMips>
  <WorkerCPUs>1</WorkerCPUs>
</WorkerPhysicalProperties>
<WorkerUsageProperties>
<WorkerTotalTime>1356</WorkerTotalTime>
<WorkerTotalWorking>1324</WorkerTotalWorking>
<WorkerTotalSuspended>32</WorkerTotalSuspended>
</WorkerUsageProperties>
```

```
</Worker>
```

## XML Requirements

- Define MW Markup Language
- Define MW Solver Markup Language Extension
- Define XML file schema (nested tree structure)
- XML visualization
  - ◊ XML simple viewer
  - ◊ XML Style Sheet script  $\rightsquigarrow$  HTML / DHTML (sent along with XML)
- Requires Internet Explorer 5 at present