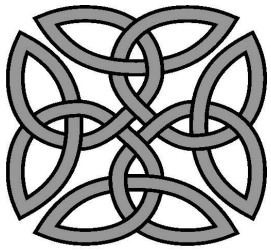


School of Mathematics

Direct Parallel Solution of Linear Systems of Dimension 10^9

Andreas Grothey, Jacek Gondzio

OOPS (Object Oriented Parallel Solver)

- Mantra: "Truly large scale problems are not only sparse but structured" (due to e.g. dynamics, uncertainty, spatial distribution etc.)
- Exploiting structure is key to building efficient IPMs for large problems:
  - Faster linear algebra
  - Reduced memory use (by use of implicit factorization)
  - Possibility to exploit (massive) parallelism
  - **We assume that structure is known!** $\Rightarrow$ no automatic detection.
- OOPS is a general purpose (parallel) IPM solver
  - Not tuned to any particular hardware
  - Not tuned to any particular problem (structure).
- OOPS currently solves LP/QP problems.
- NLP extension solves nonlinear financial planning problems

Interior Point Methods

$$\min c^\top x + \tfrac{1}{2} x^\top Q x \quad \text{s.t.} \quad \begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned} \qquad \text{(QP)}$$

Optimality conditions:

$$\begin{aligned} c + Qx - A^\top y - z &= 0 \\ Ax &= b \\ XZe &= 0 \quad (\mu e) \\ x, z &\geq 0 \end{aligned}$$

$\Rightarrow$ Newton Step:

$$\begin{bmatrix} -Q - \Theta & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix} := \begin{bmatrix} Qx + c - A^\top y - \mu X^{-1} e \\ b - Ax \end{bmatrix} \qquad \text{(NS-QP)}$$

where

$$\Theta = X^{-1} Z, \quad X = \operatorname{diag}(x), \quad Z = \operatorname{diag}(z)$$

Linear Algebra of IPMs

Main work: solve

$$\underbrace{\begin{bmatrix} -Q-\Theta & A^\top \\ A & 0 \end{bmatrix}}_{\Phi} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}$$

for several right-hand-sides at each iteration

$\Rightarrow$ Two stage solution procedure

- factorize $\Phi = LDL^\top$
- backsolve(s) to compute direction $(\Delta x, \Delta y)$ + corrections

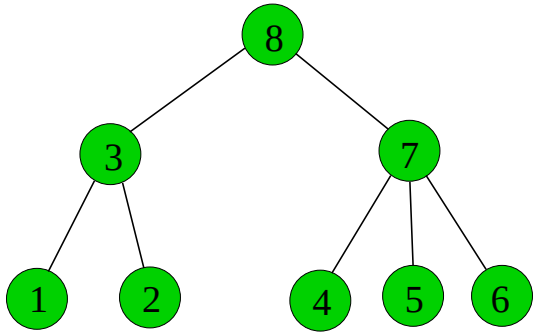
$\Rightarrow$ Φ changes numerically but not structurally at each iteration

Key to efficient implementation is exploiting structure of Φ in these two steps

OOPS: (Block) Elimination Trees

Elimination tree orders rows/columns for elimination with minimum fill-in:

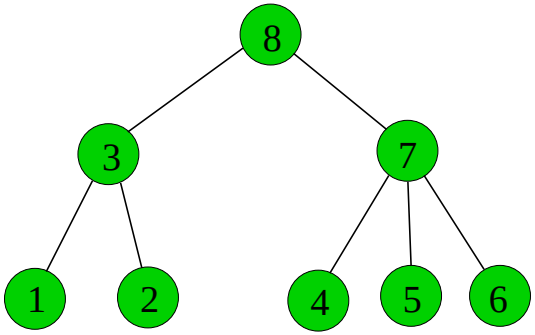
$$\begin{bmatrix}
x & & x & & & & & x \\
 & x & x & & & & & x \\
x & x & x & & & & & x \\
 & & & x & & & x & x \\
 & & & & x & & x & x \\
 & & & & & x & x & x \\
 & & & x & x & x & x & x \\
x & x & x & x & x & x & x & x
\end{bmatrix}$$



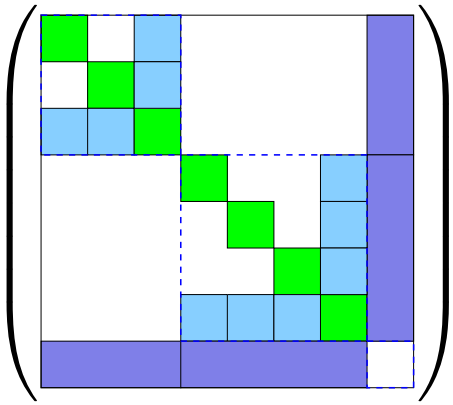
OOPS: (Block) Elimination Trees

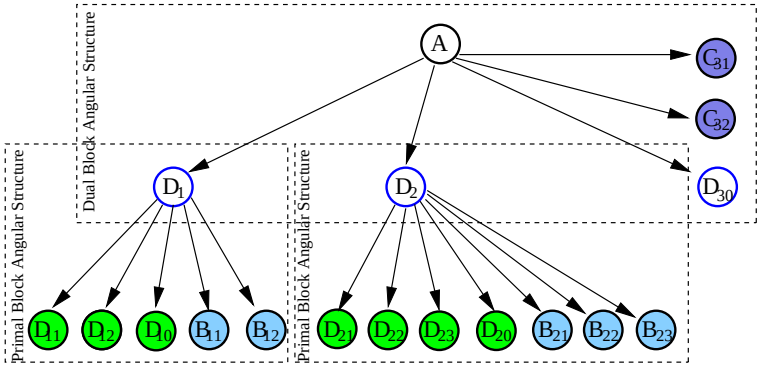
Elimination tree orders rows/columns for elimination with minimum fill-in:

$$\begin{bmatrix} x & & x & & & & & x \\ & x & x & & & & & x \\ x & x & x & & & & & x \\ & & & x & & & x & x \\ & & & & x & & x & x \\ & & & & & x & x & x \\ & & & x & x & x & x & x \\ x & x & x & x & x & x & x & x \end{bmatrix}$$



Elimination Tree can be extended to Block Elimination Tree

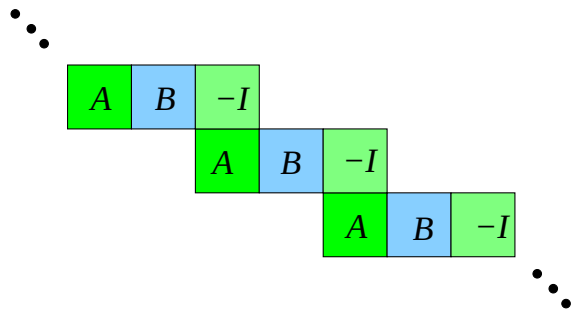




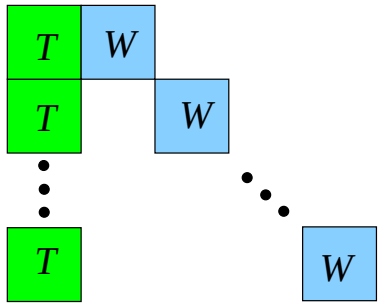
$\Rightarrow$ Organisation of linear algebra, Parallelism

Sources of Structure

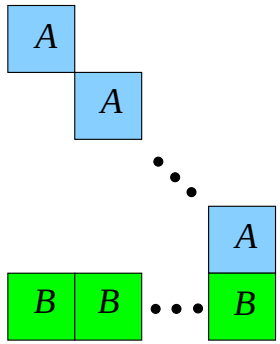
(linear) dynamics: $x_{t+1} = Ax_t + Bu_t$



uncertainty:
common 1st stage decision (today)
recourse action in 2nd stage (tomorrow)
$Tx + W_i\, y_i = h_i$

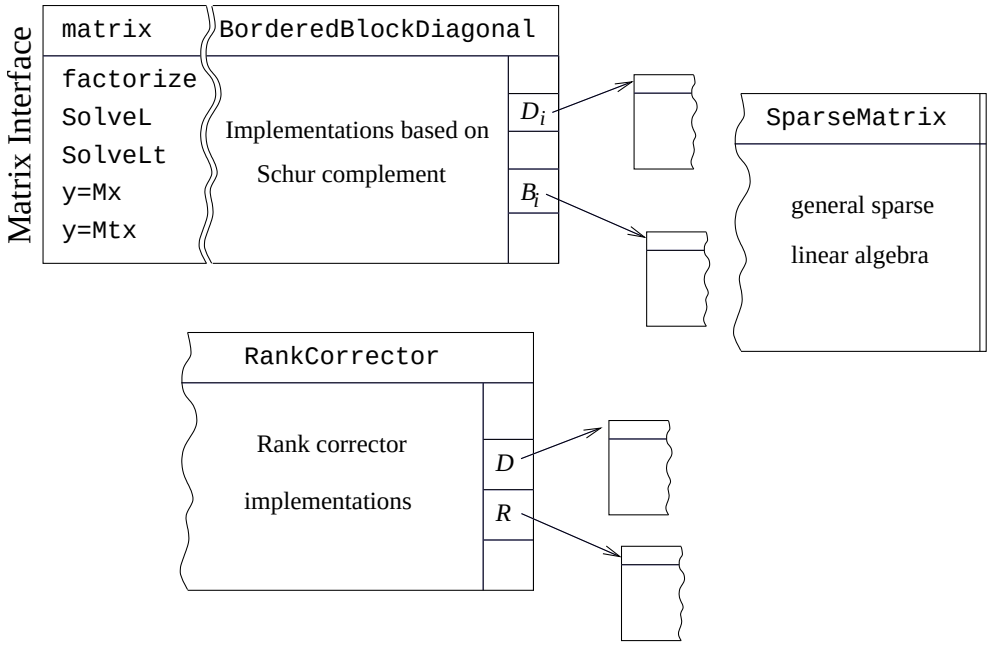


almost "independent" divisions
share a **common resource**
$\sum_{i=1}^{k} B_i\, x_i = b$



OOPS: Object-oriented linear algebra implementation

- Every node in *block elimination tree* has own linear algebra implementation (depending on its type)
- Implementation is realisation of an abstract linear algebra interface.
- Different implementations for different structures are available.



$\Rightarrow$ Rebuild *block elimination tree* with matrix interface structures

Application: Financial Planning (ALM) I

- A set of assets $\mathcal{J} = \{1, ..., J\}$ is given (e.g. bonds, stock, real estate).
- At every stage $t = 0, ..., T-1$ we can buy or sell different assets.
- The return of asset j at stage t is *uncertain* (but distribution is known).

We have to make investment decisions: what to buy or sell, at which time stage

Objectives:

- maximize the final wealth
- minimize the associated risk

$\Rightarrow$ Mean Variance formulation:
$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$

Possible extensions (lead to nonlinear models)

- different risk measures (one-sided, expected shortfall/value at risk)
- (nonlinear) utility functions

Application: Financial Planning (ALM) II

Stochastic Programming approach to Financial Planning

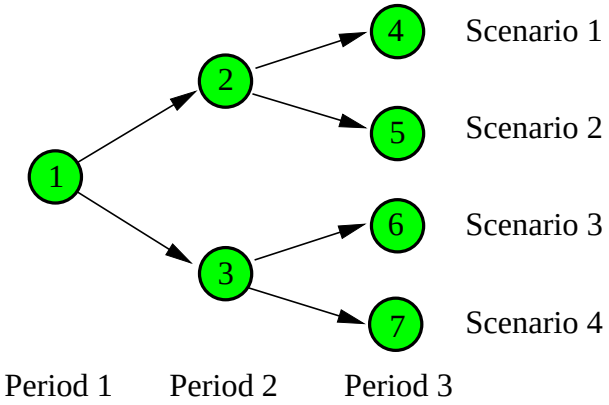
- Popular with academics, practitioners seem suspicious
  - Large problem sizes
  - Places constraints on model (?)
- Traditionally solved by decomposition
  - Nested Benders Decomposition aka L-shaped method
  - Works well for LP models
  - Unclear how to extend to quadratic/nonlinear models

$\Rightarrow$ Decompose the linear algebra

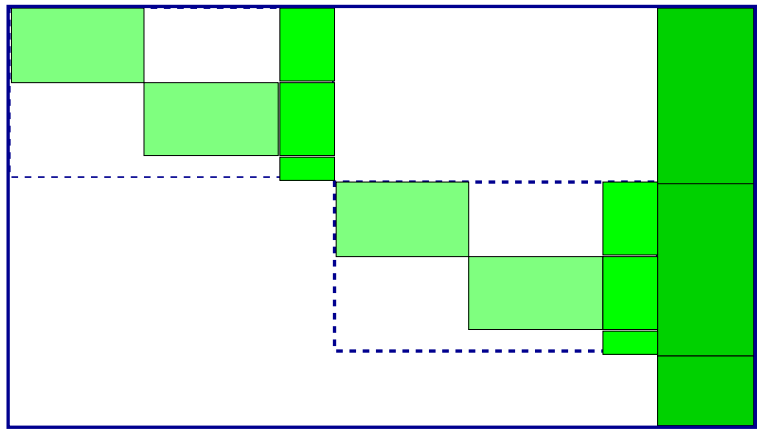
New application for massively parallel architectures

Although 5% of top500 GFlops are in Finance

Multistage Stochastic Programming



Scenario Tree

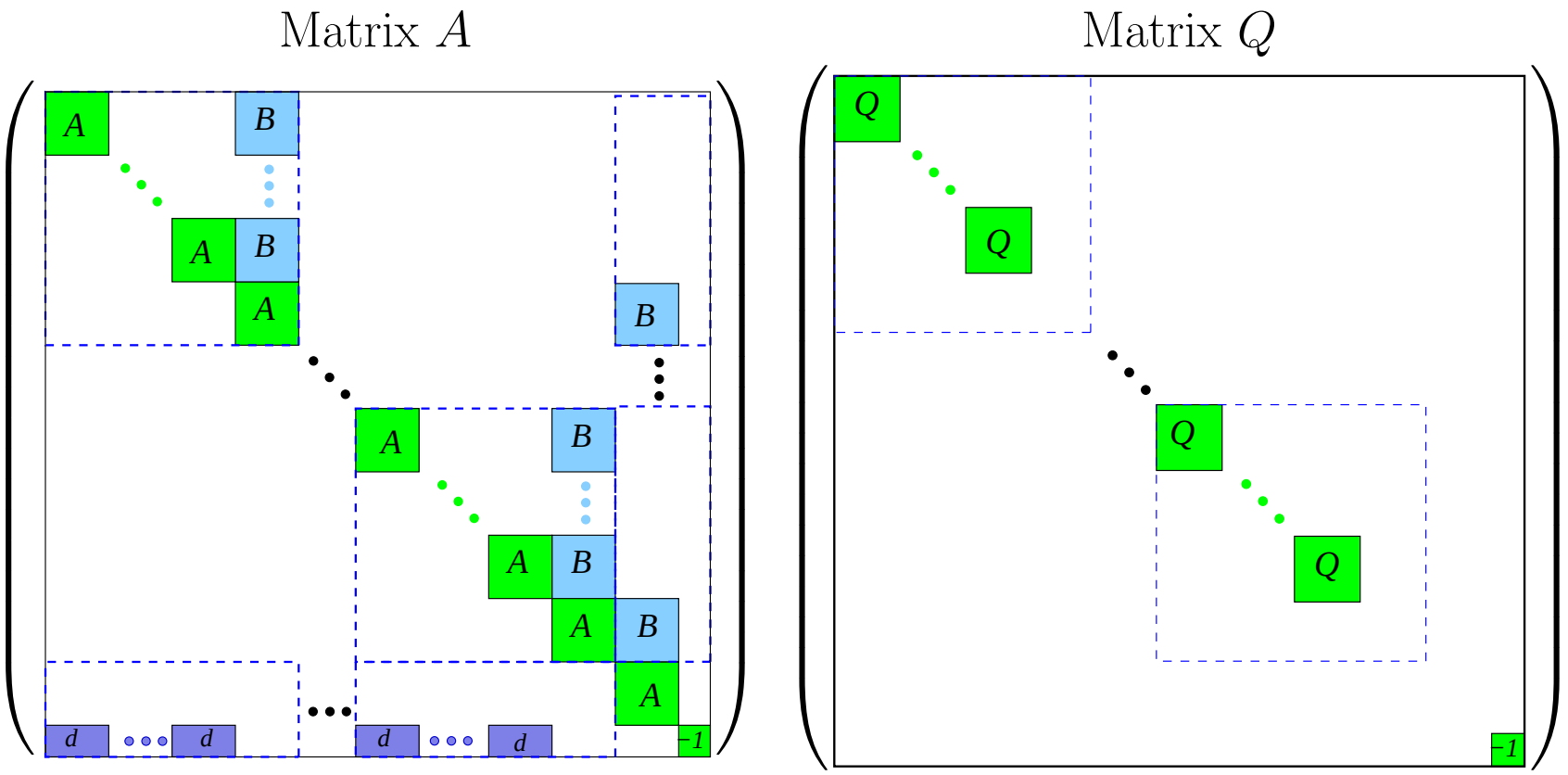


Constraint Matrix

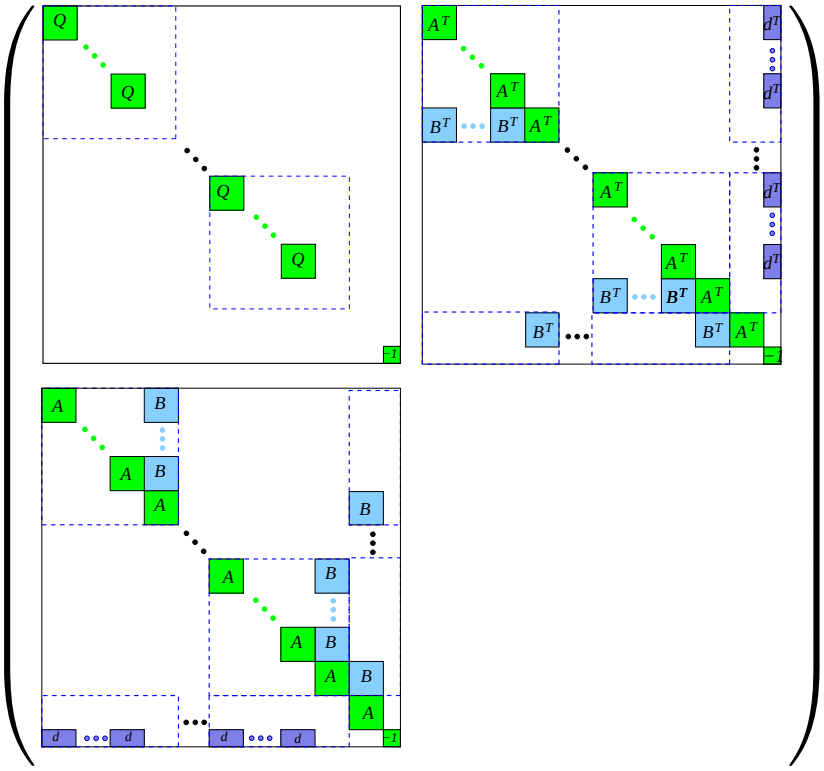
Symmetrical event tree with p realizations at each node and T periods corresponds to

p^{T-1} scenarios $\qquad$ $\dfrac{p^T-1}{p-1}$ nodes (blocks)

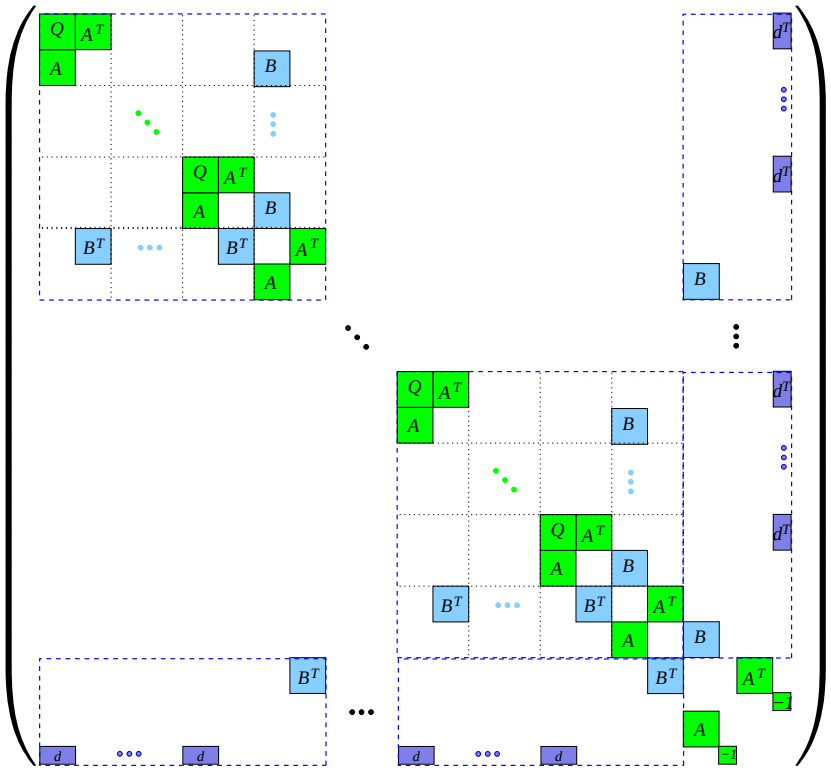
ALM: Structure of matrices A and Q:



Structures of A and Q imply structure of Φ



$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix}$$



$$P \begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} P^{-1}$$

ALM: Problem Size

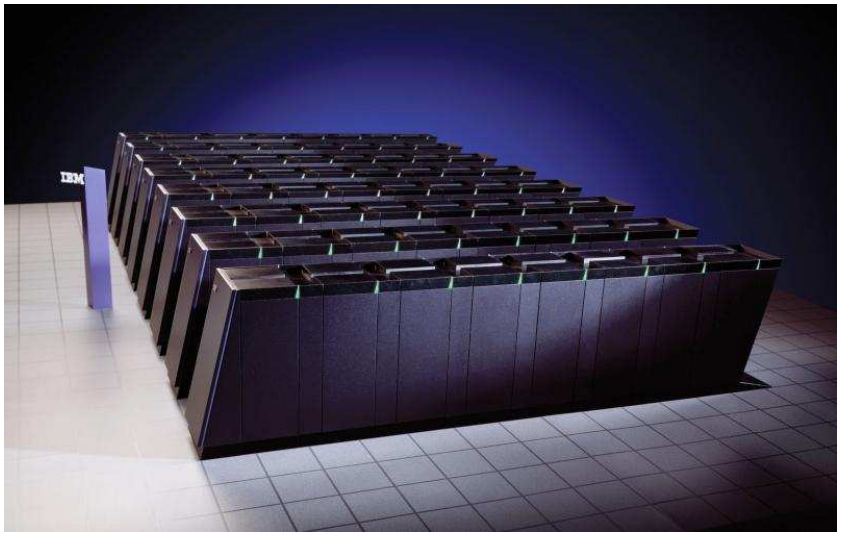
Optimization of 21 assets (stock market indices).
$\Rightarrow$ Scenario tree needs to capture

- correlations of 21 assets
- correlations over time periods

$\Rightarrow$ 100 branches over 6 stages $= 10^{12}$ scenarios

Better:

- Scenario tree geometry: 128-30-16-10-5-4 $\Rightarrow$ 16 million scenarios.
- Scenario Tree generated using geometric Brownian motion.
- $\Rightarrow$ 1.01 billion variables, 353 million constraints



BlueGene/L (Edinburgh, Scotland)

- 2048 Processors
- 0.7GHz, 256Mb
- 4.7 TFlops
- #81 in `top500.org` list

HPCx (Daresbury, England)

- 1600 IBM Power-4 Processors
- 1.7GHz, 800Mb
- 6.2 TFlops
- #57 in `top500.org` list

Issues for Massive Parallelism

- Bootstrapping
- Sparsity of multilevel linear algebra
- Memory management

Bootstrapping

Problem: Data for the problem is $\approx$ 25 GB. We have 512MB/node on BlueGene.
How to get the data on the processors ?

Data can be represented as

- core: 64 variables, 22 constraints
- scenario tree (topology, probability, returns for each node): 2.6 GB

Still does not fit on one processor

- Scenario tree data (probability, returns) can be generated in parallel
- Scenario tree topology needs $\approx$ 30 MB

This is all that OOPS needs to decide on the allocation of blocks to processors

Bootstrapping II

1. Read problem dimensions,
   generate scenario tree topology and matrix tree (30MB)
2. Call OOPS: Divide matrix tree among processors,
   set up data structures on each processor
3. Generate scenario tree data on every processor
4. On every processor: Call-back to generator to fill in SparseMatrix data
5. Start solution process

Example: Bordered Block-Diagonal Structure (Schur complement)

$$\underbrace{\begin{pmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_0 \end{pmatrix}}_{\Phi} = \underbrace{\begin{pmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{1,0} & \cdots & L_{n,0} & L_0 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} D_1 & & & \\ & \ddots & & \\ & & D_n & \\ & & & D_0 \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} L_1^\top & & & L_{1,0}^\top \\ & \ddots & & \vdots \\ & & L_n^\top & L_{n,0}^\top \\ & & & L_0^\top \end{pmatrix}}_{L^\top}$$

- Cholesky-like factors can be obtained by Schur-complement:

$$\begin{aligned} \Phi_i &= L_i D_i L_i^\top & L_{i,0} &= B_i L_i^{-\top} D_i^{-1}, \ i = 1, \ldots n \\ C &= \Phi_0 - \textstyle\sum_{i=1}^n L_{i,0} D_i L_{i,0}^\top & C &= L_0 D_0 L_0^\top \end{aligned}$$

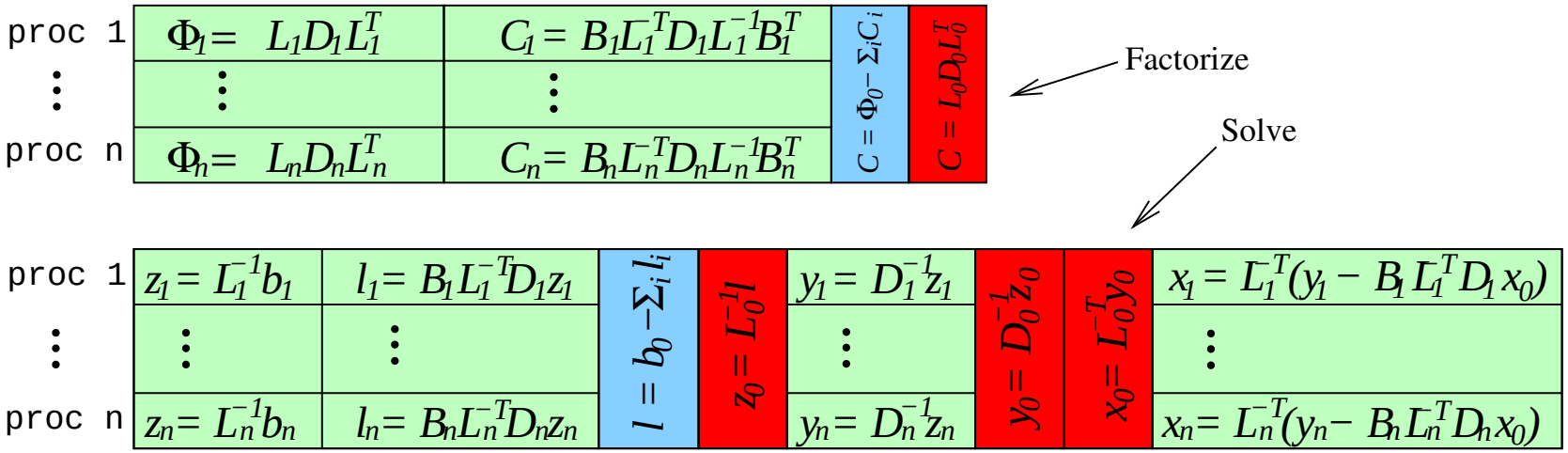
- And the system $\Phi x = b$ can be solved by

$$\begin{aligned} z_i &= L_i^{-1} b_i \\ z_0 &= L_0^{-1}(b_0 - \textstyle\sum L_{i,0} z_i) \\ y_i &= D_i^{-1} z_i \end{aligned} \qquad \begin{aligned} x_0 &= L_0^{-\top} y_0 \\ x_i &= L_i^{-\top}(y_i - L_{i,0}^\top x_0) \end{aligned}$$

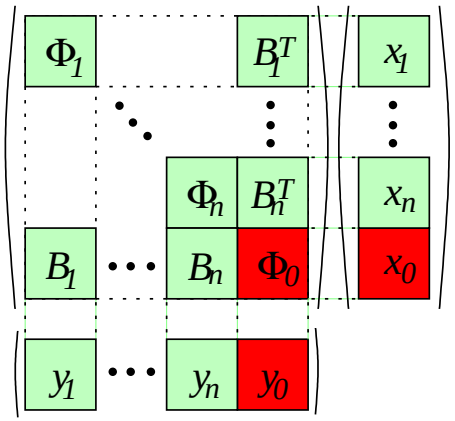
- Operations (Cholesky, Solve, Product) are only performed on sub-blocks
  $\Rightarrow$ Can also exploit structure in sub-blocks

Exploiting Parallelism: Bordered Block-Diagonal Structure:

- Distribution of computations:



- Storage:



processor 1

processor n

all processors

On separate processors

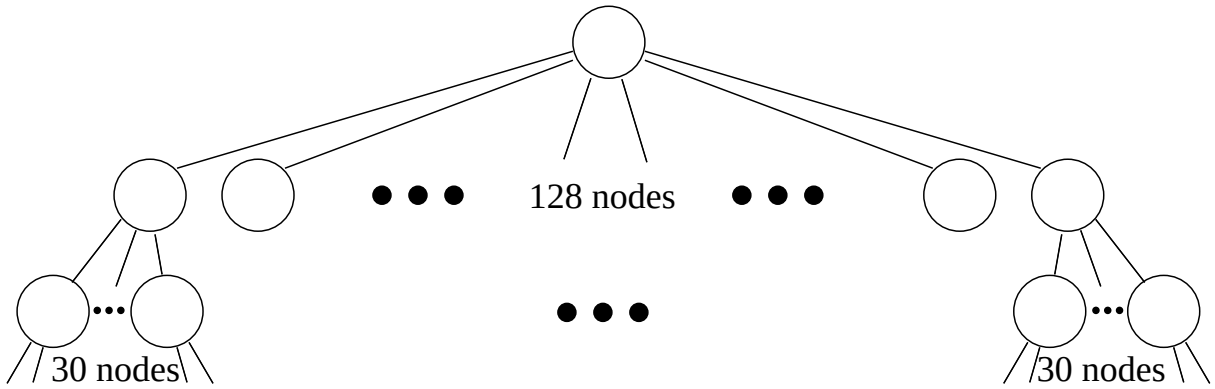
On all processors

Communications

Parallel division of the problem

- In ALM problems matrices up to $\approx 500.000 - 1.000.000$ variables can be treated as unstructured sparse matrices on one processor
- Problem has:
  - 128 first level nodes with 10.000.000 variables each.
  - 3840 second level nodes with 350.000 variables each.

$\Rightarrow$ need to decompose problem at second level
(with 1280 processors $\Rightarrow$ 3 blocks per processor)

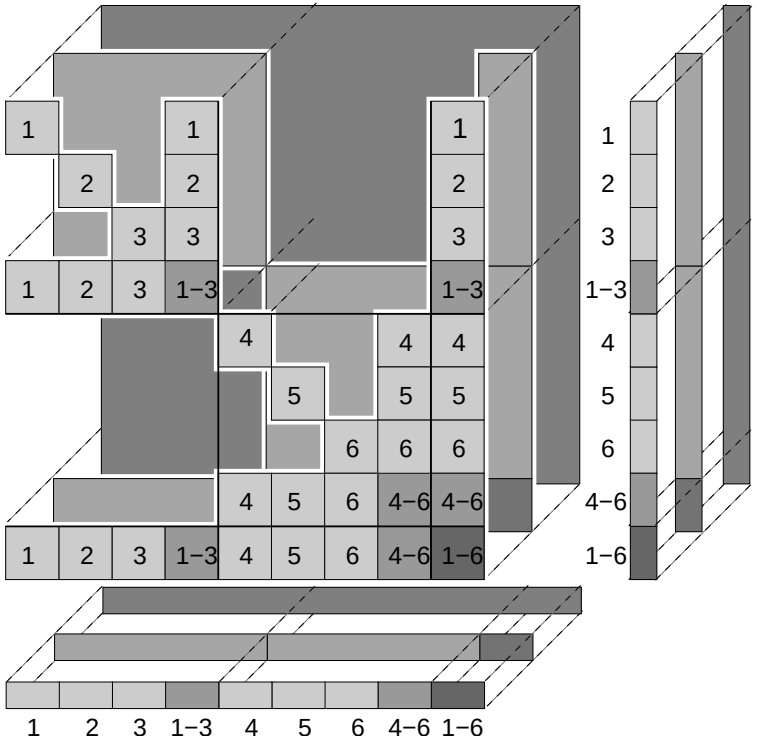


Memory Management

- Data for problem requires 2.6GB of memory.
  $\Rightarrow$ need to split information between processors
- To each node in block-elimination tree a set of processors is assigned
- Linear Algebra is implemented so that processors communicate when needed

Distribution of leading matrix blocks among processors implies

- Distribution of subordinate blocks
- Distribution of row/column vector contributions



Results

Stages	Blk	Assets	Scenarios	Constraints	Variables	iter	time	procs	machine
7	128	6	12,831,873	64,159,366	153,982,477	42	3923	512	BG/L
7	64	14	6,415,937	96,239,056	269,469,355	39	4692	512	BG/L
7	128	13	12,831,873	179,646,223	500,443,048	45	6089	1024	BG/L
7	128	21	16,039,809	352,875,799	1,010,507,968	53	3020	1280	HPCx

Parallel Efficiency (strong scaling)

nodes	peak Mem	time	Comm	Cholesky	Solves	MatVectProd
16	426MB	2587 (1.00)	24	1484 (1.00)	956 (1.00)	28.8 (1.00)
32	232MB	1303 (0.99)	13	743 (1.00)	485 (0.98)	18.0 (0.80)
64	132MB	688 (0.94)	6	377 (0.98)	270 (0.88)	13.0 (0.55)
128	84MB	348 (0.93)	3	187 (0.99)	139 (0.86)	9.0 (0.40)
256	56MB	179 (0.90)	3	93 (0.99)	73 (0.82)	5.8 (0.31)
512	46MB	94 (0.86)	2	47 (0.98)	39 (0.76)	3.9 (0.23)

Parallel Efficiency (weak scaling)

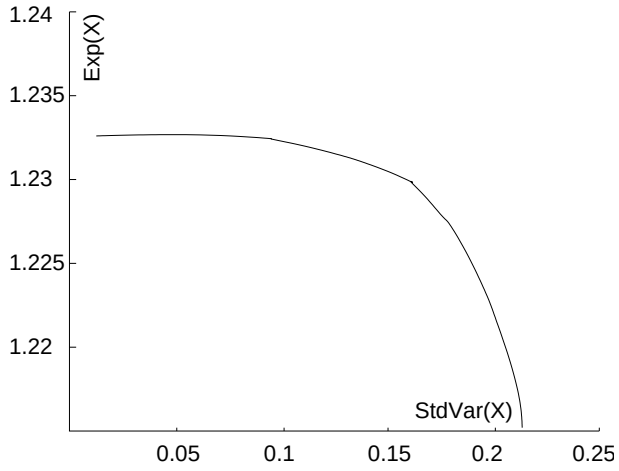
Prob	f.s.d.	vars	cons	nz(A)	nz(L)	peak Mem
A16	2	2,806,987	7,819,462	15,638,922	118,774,704	260MB
A32	4	5,613,959	15,638,884	31,277,766	237,549,408	260MB
A64	8	11,227,903	31,277,728	62,555,454	475,098,816	264MB
A128	16	22,455,791	62,555,416	125,110,830	950,197,632	264MB
A256	32	44,911,567	125,110,792	250,221,582	1,900,395,264	268MB
A512	64	89,823,133	250,221,582	500,443,086	3,800,790,528	276MB
A1024	128	179,646,223	500,443,048	1,000,886,094	7,601,581,056	292MB

Prob	nodes	time(20iter)	peak mem/node	generation	communication	rest
A16	16	1815	260MB	6	26	1783
A32	32	1845	260MB	12	51	1782
A64	64	1911	264MB	23	102	1786
A128	128	2050	264MB	45	206	1799
A256	256	2289	268MB	89	416	1784
A512	512	2797	276MB	178	825	1794
A1024	1024	3818	294MB	361	1666	1791

Results (Efficient Frontier)

Mean Variance formulation:

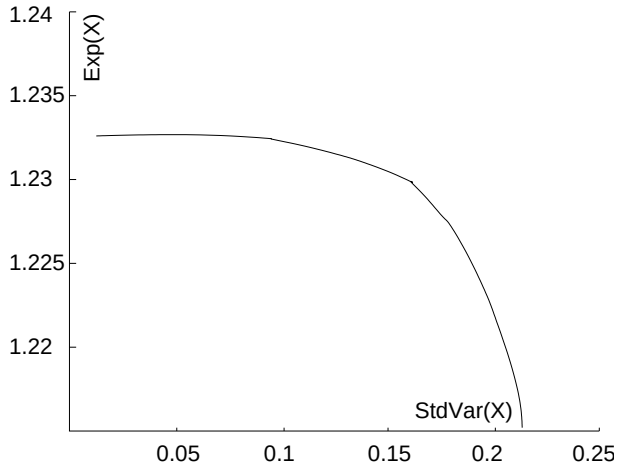
$$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$$



Results (Efficient Frontier)

Mean Variance formulation:

$$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$$



constraints	variables	np	ρ =0.001	0.005	0.01	0.05	0.1	0.5	1	5	10
223,321	76,881	1	14	14	14	14	14	13	17	16	17
			14	5	5	5	4	5	5	8	8
533,725	198,525	1	14	14	14	14	14	15	18	18	17
			14	5	5	5	6	5	5	9	10
5,982,604	16,316,191	32	24	23	24	23	25	22	24	23	24
			24	8	11	13	11	13	12	12	14
70,575,308	192,478,111	512	52	53	45	43	44	42	44	46	46
			52	13	13	15	15	16	16	23	25

Conclusions:

- OOPS is a general purpose IPM solver with object-oriented linear algebra
- Will scale up to massively parallel architecture
- Can solve problems with 10^9 variables using direct factorization methods
- Allows the solution of realistic financial planning problems

Object-Oriented Parallel Solver (OOPS):

`http://www.maths.ed.ac.uk/~gondzio/parallel/solver.html`

References:

- J. Gondzio and A. Grothey, *Parallel interior point solver for structured quadratic programs: application to financial planning problems*, Tech. Rep. MS-03-001, School of Maths, University of Edinburgh, April 2003 (to appear in Annals of OR).
- J. Gondzio and A. Grothey, *Solving nonlinear portfolio optimization problems with the primal-dual interior point method*, Tech. Rep. MS-04-001, School of Maths, University of Edinburgh, May 2004 (to appear in EJOR).
- J. Gondzio and A. Grothey, *Exploiting Structure in Parallel Implementation of Interior Point Methods for Optimization*, Tech. Rep. MS-04-004, School of Maths, University of Edinburgh, Dec 2004.
- J. Gondzio and A. Grothey, *Direct Solution of Linear Systems of Size* 10^9 *Arising in Optimization with Interior Point Methods*, Tech. Rep. MS-05-003, School of Maths, University of Edinburgh, Nov 2005 (to appear in Parallel Processing and Applied Mathematics, Lecture Notes in Computer Sciences 3911).