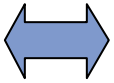
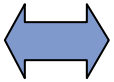
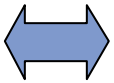
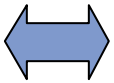
Introduction to Domain Decomposition: Context, Basic Linear Algorithms, Convergence, and Scaling

David E. Keyes

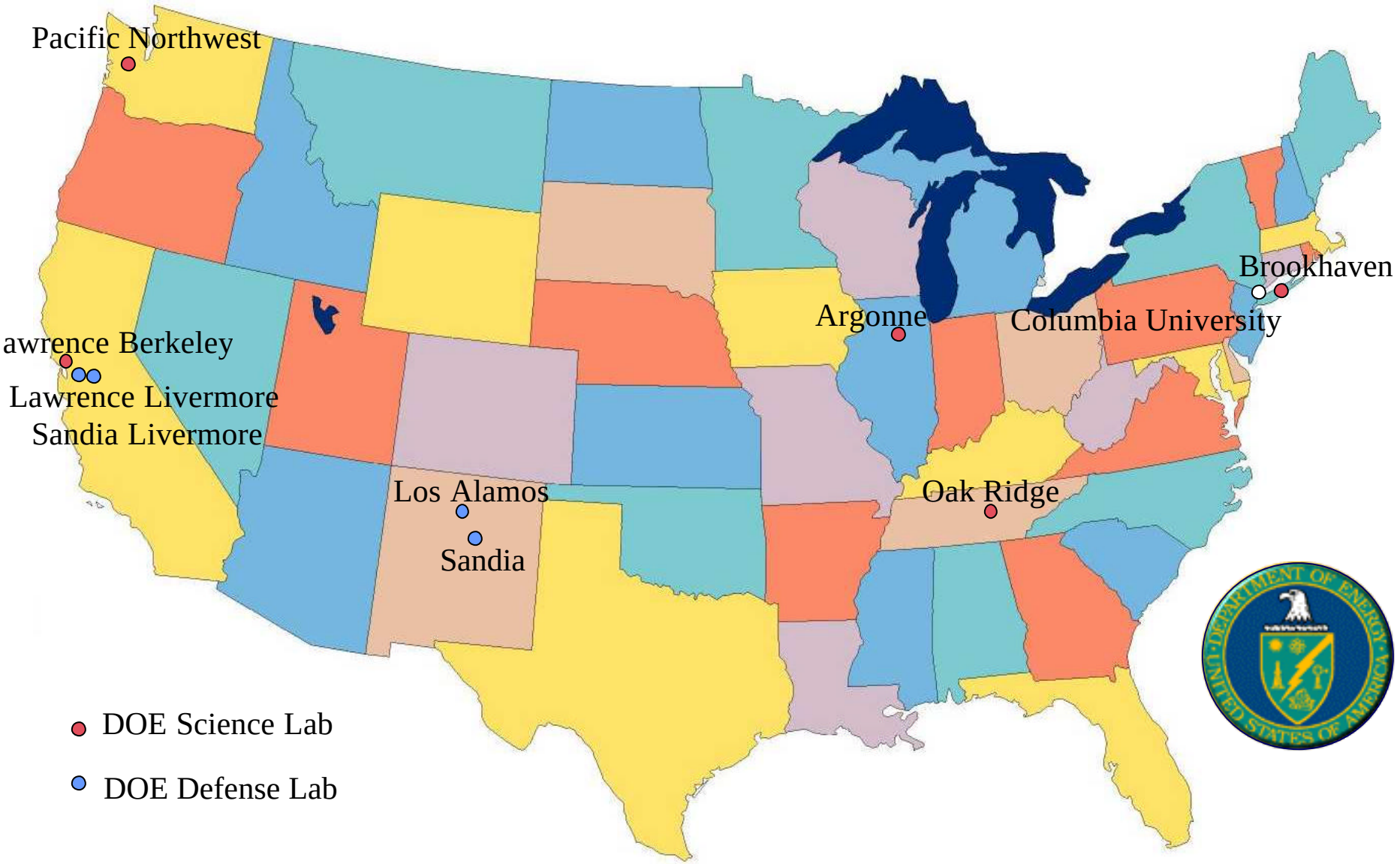
Department of Applied Physics & Applied Mathematics
Columbia University

Institute for Scientific Computing Research
Lawrence Livermore National Laboratory

Who is this guy up front?

- B.S.E., Aerospace and Mechanical Sciences, Princeton 1974-1978
- M.S. & Ph.D., Applied Mathematics, Harvard, 1978-1984
- Post-doc (under W. D. Gropp), Computer Science, Yale, 1984-1985
- Asst. and Assoc. Prof., Mechanical Engineering, Yale, 1986-1993  Consultant, ICASE, NASA Langley, 1986-1993
- Assoc. Prof., Computer Science, Old Dominion, 1993-1999  Associate Research Fellow, ICASE, NASA Langley, 1993-2002
- Professor, Mathematics & Statistics, Old Dominion, 1999-2003  Acting Director, ISCR, Lawrence Livermore, 1999-
  - Chair, 1999-2001
  - Director, Center for Computational Science, 2001-2003
- Professor, Applied Physics & Applied Mathematics, Columbia, 2003-  Faculty Associate, CDIC, Brookhaven, 2003-

Major U. S. DOE labs



How does this lecture series fit in?

- **Compared to Gropp's lectures on *what* is in PETSc and *how* to use PETSc …**

  … these lectures describe *why* certain algorithms are in PETSc.

- **Compared to the *careful, systematic, theoretical* approach of a mathematician …**

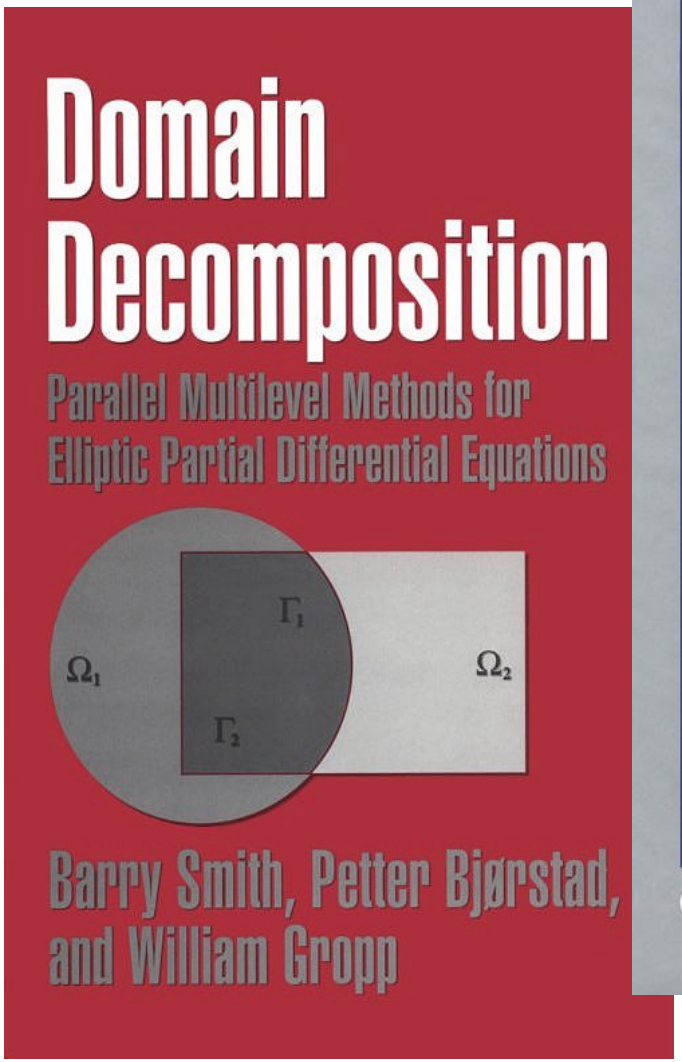
  … these lectures are *practical*, rely on *intuition*, and *defer missing details* to other sources

Overall series and on-line resources

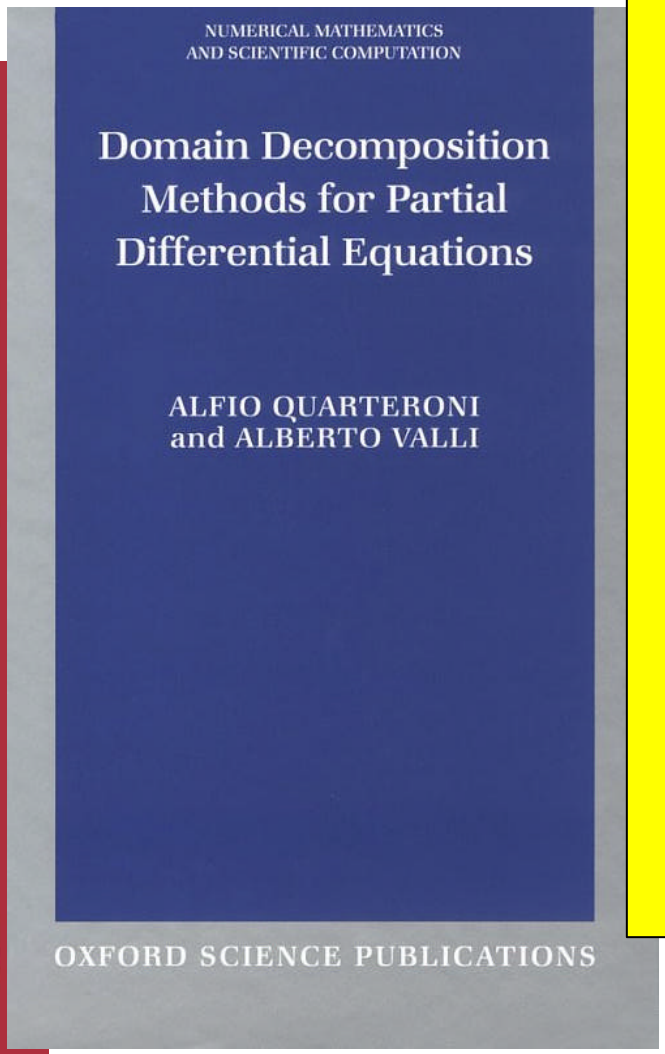
- **Four lectures on:**
  - **Introduction and DD for basic linear problems**
  - **Nonlinear and transient problems**
  - **Examples of advanced applications**
  - **Physics-based preconditioning and optimization**
- **Roughly coordinated with W. D. Gropp's**
- **See DD-15 website for recently posted links to publicly available pdfs of three papers each per lecture**

Resources for deeper study

1992



1997



Domain Decomposition Methods

O. B . Widlund and A. Toselli

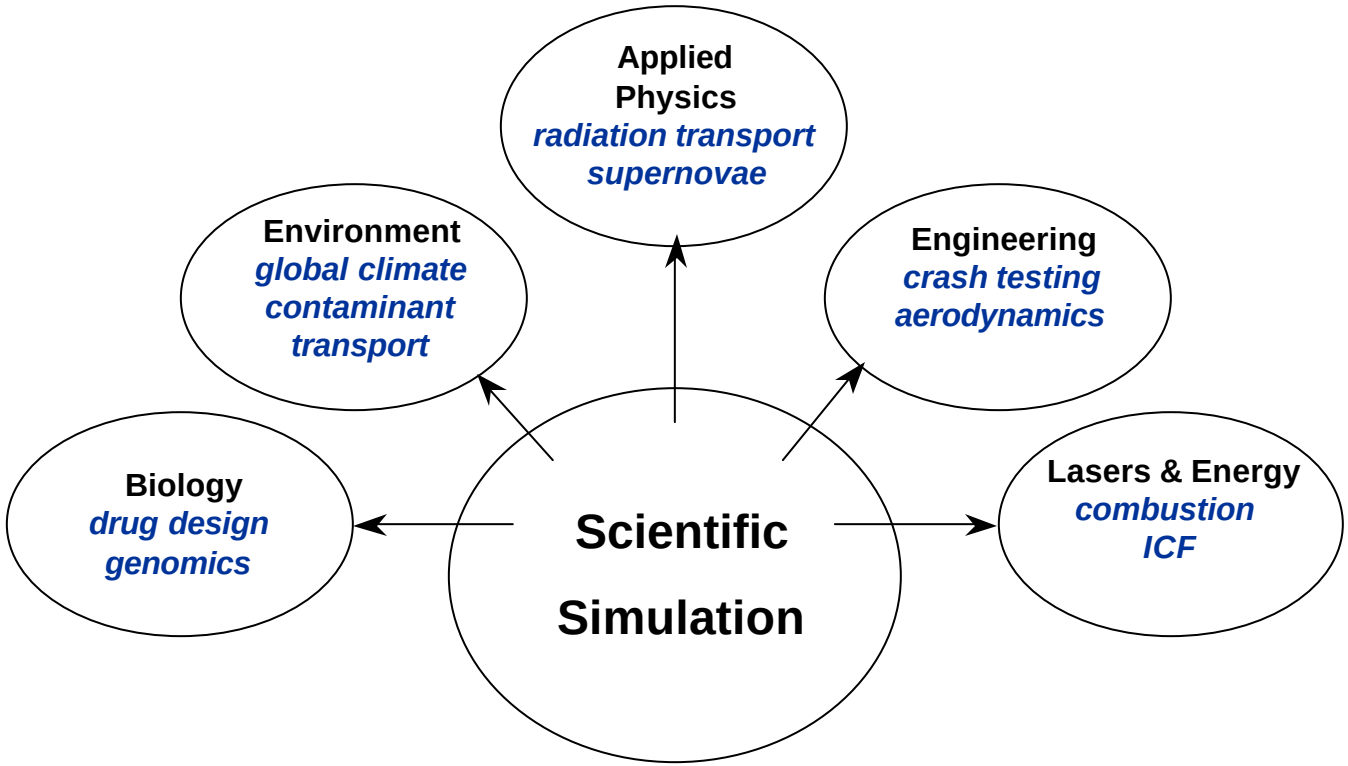
2003?

See also famous SIAM Review paper by Xu, 1992

Plan of presentation

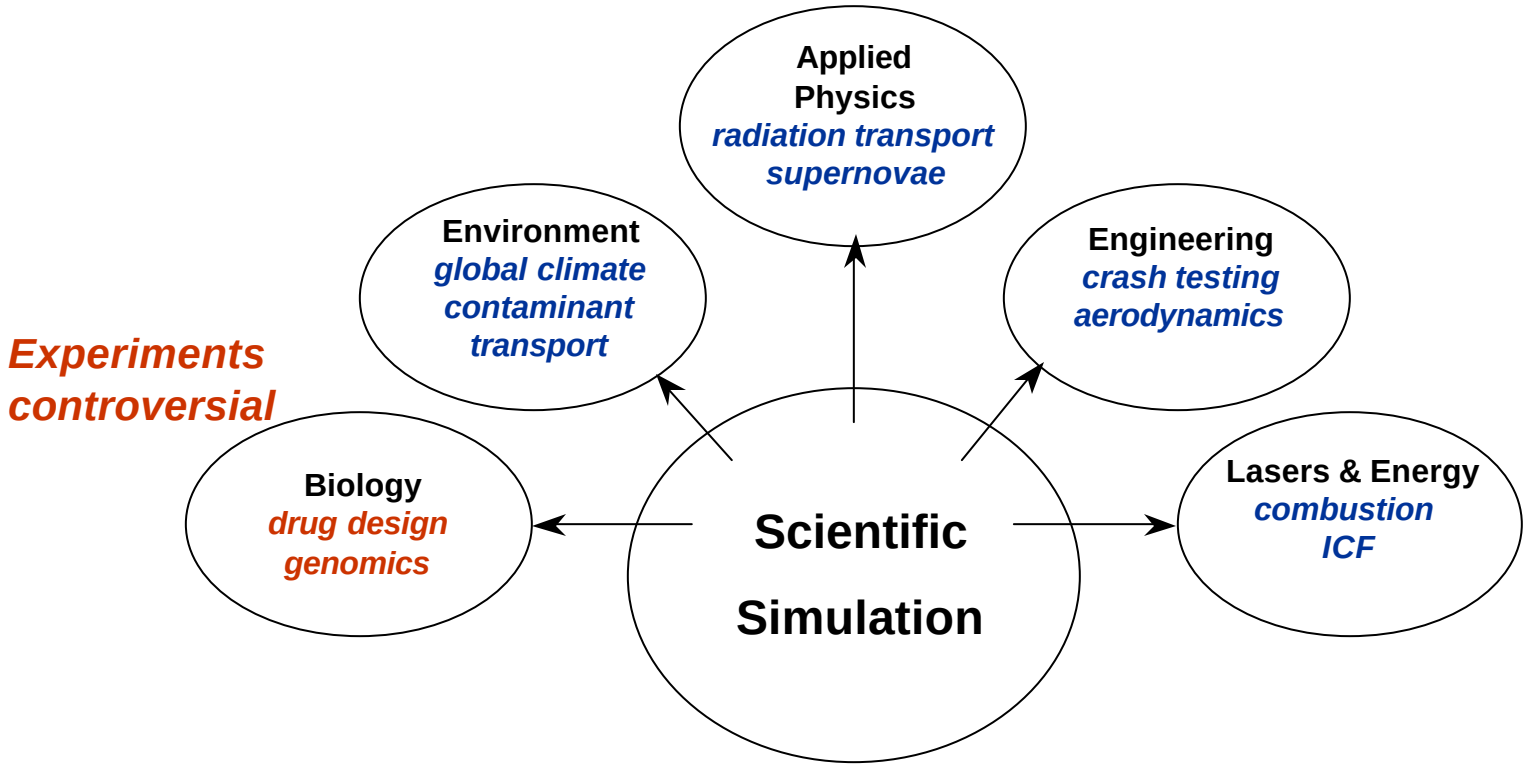
- **Imperative of domain decomposition (DD) for terascale computing**
  - **from viewpoint of architecture**
  - **from applications (more on this Friday AM)**
- **Basic DD algorithmic concepts**
  - **Schwarz**
  - **Schur**
  - **Schwarz-Schur combinations**
- **Basic DD convergence and scaling properties**

Terascale simulation has been “sold”



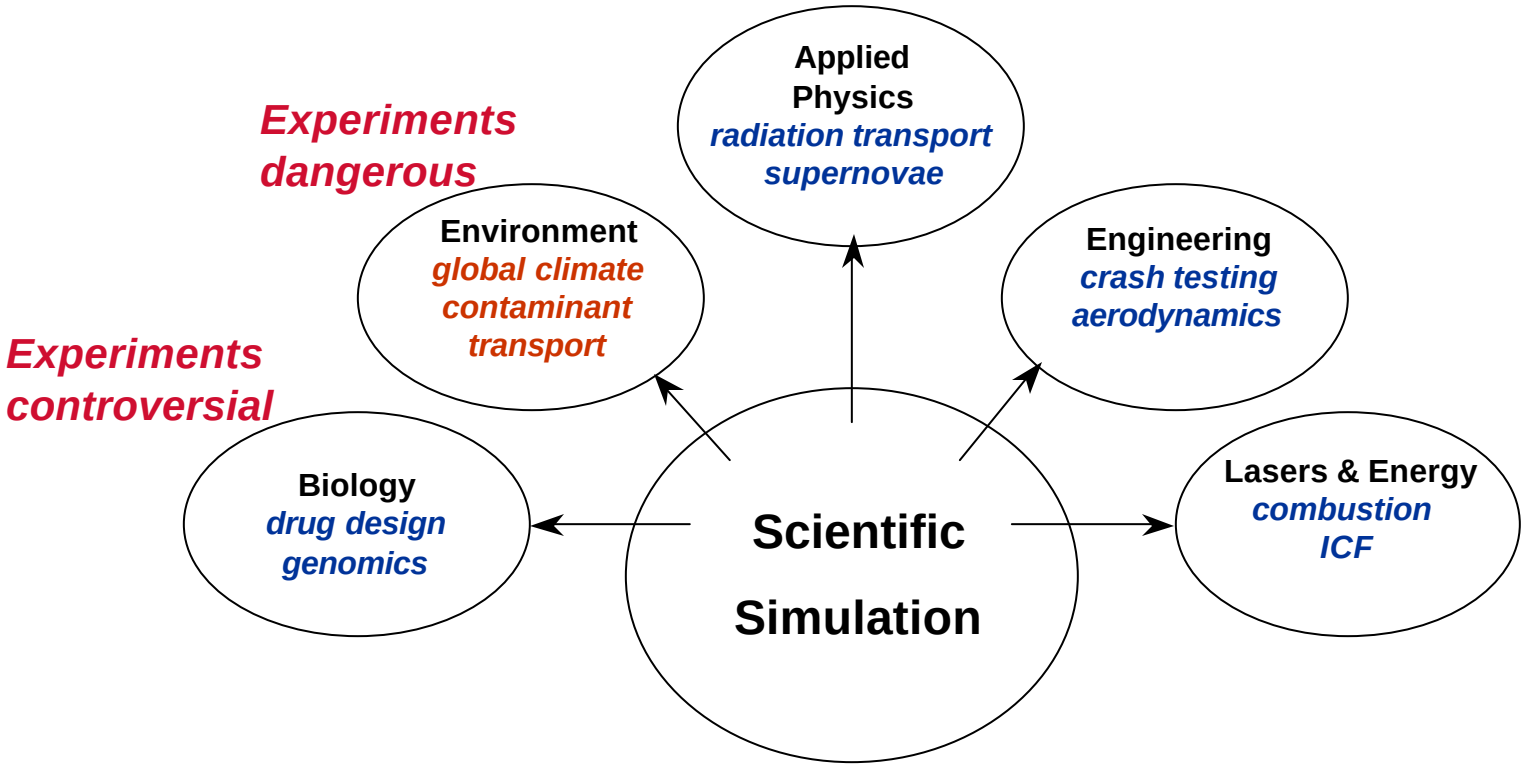
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been “sold”



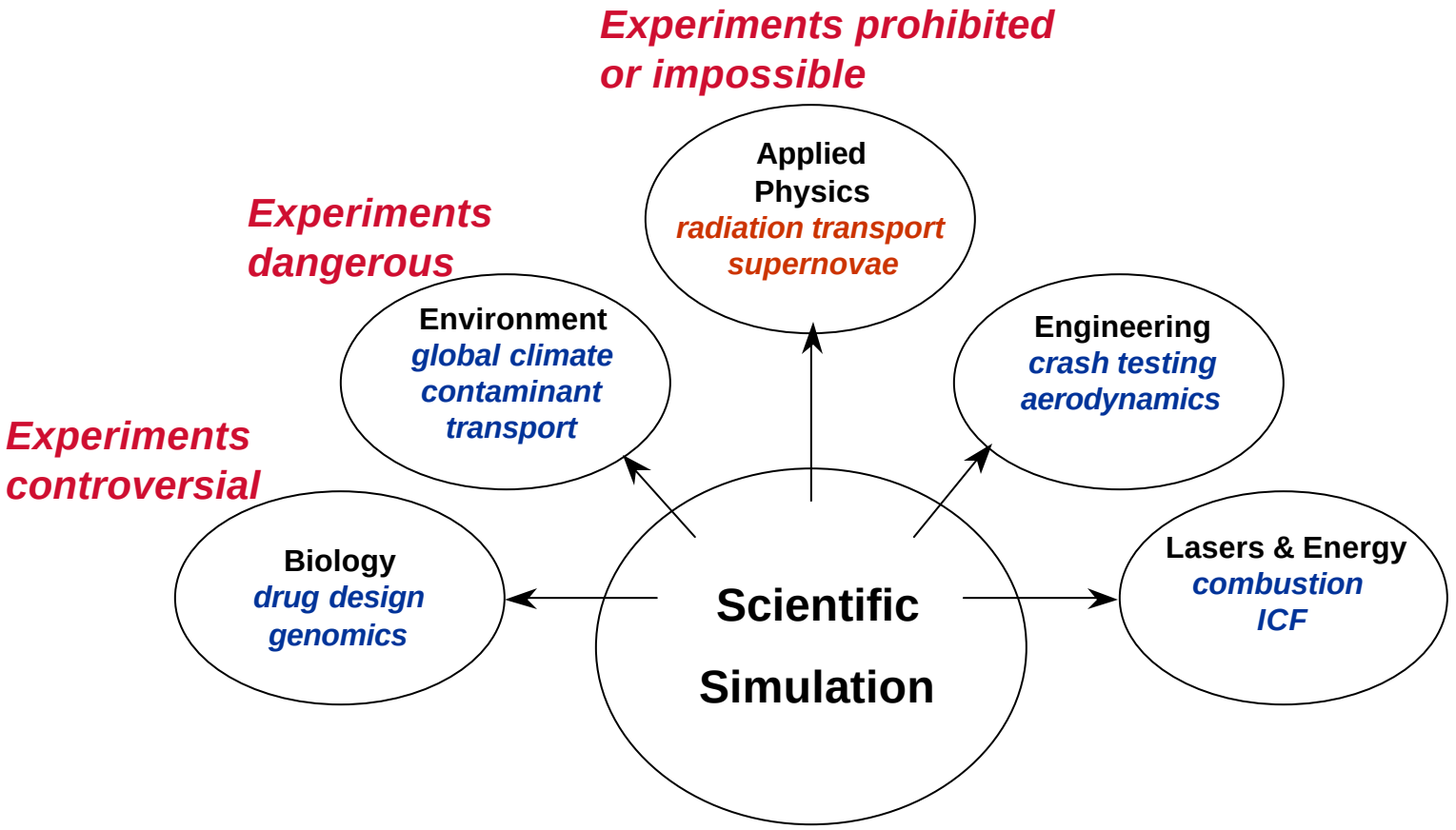
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been “sold”



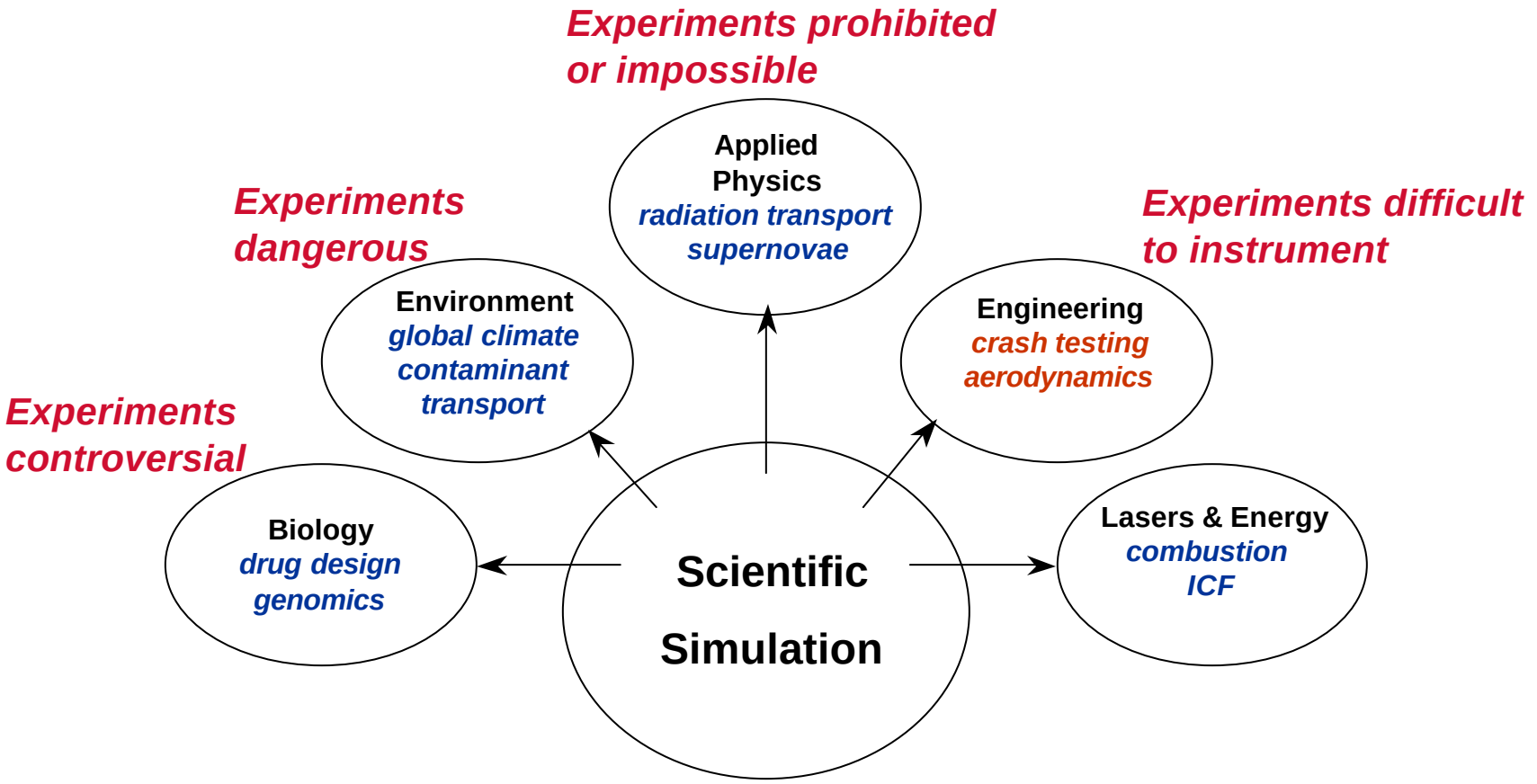
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"



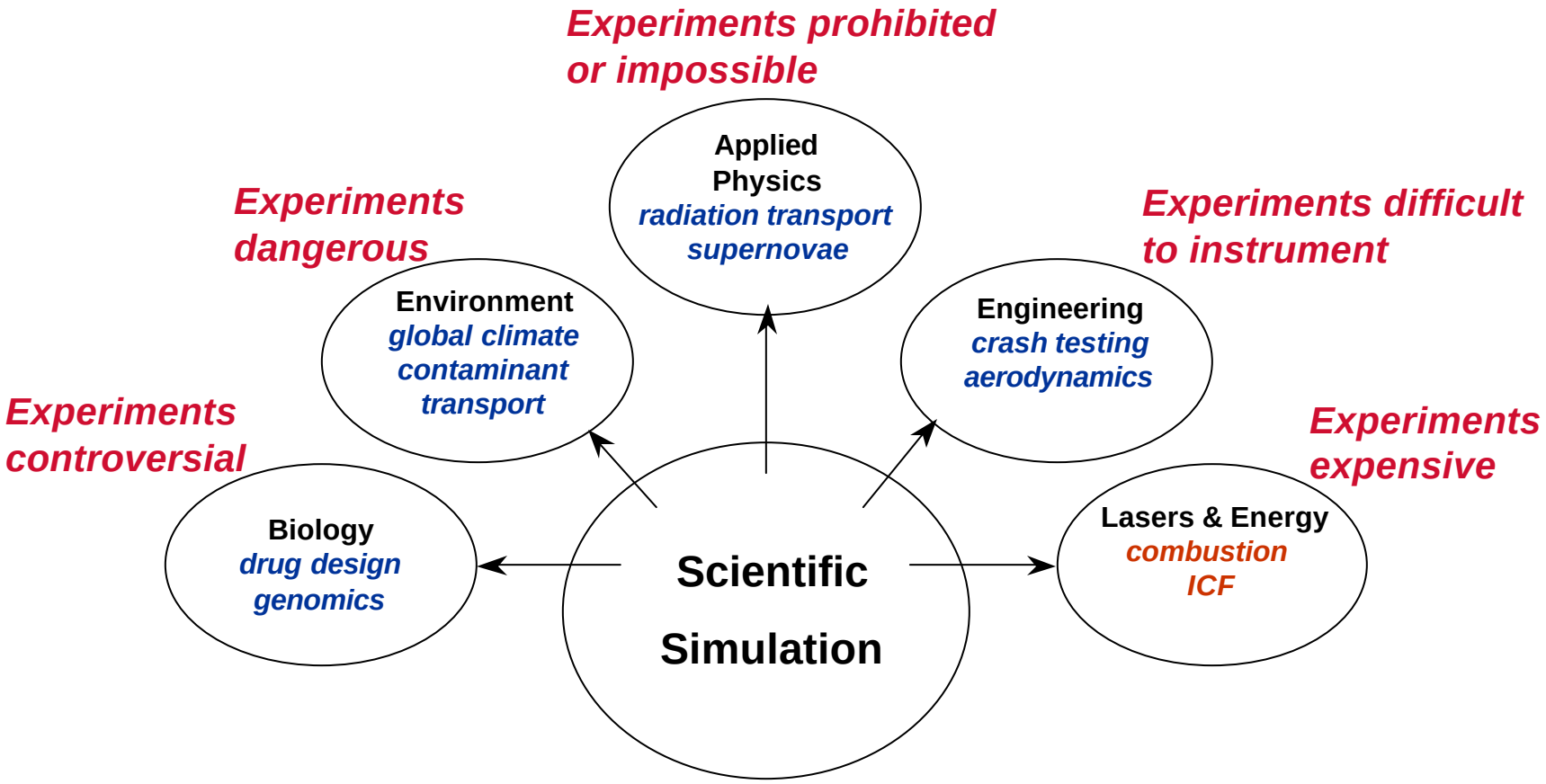
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been “sold”



In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"



In these, and many other areas, simulation is an important complement to experiment.

Large platforms provided for ASCI

- ASCI roadmap is to go to 100 Teraflop/s by 2006
- Use variety of vendors
  - Compaq
  - Cray
  - Intel
  - IBM
  - SGI
- Rely on commodity processor/memory units, with tightly coupled network
- Massive software project to rewrite physics codes for distributed shared memory


NPACI
Blue Horizon


ASCI White


ASCI Q


ASCI
Blue Mountain


ASCI Cplant


ASCI Blue-Pacific


ASCI
Whitecap


ASCI Red

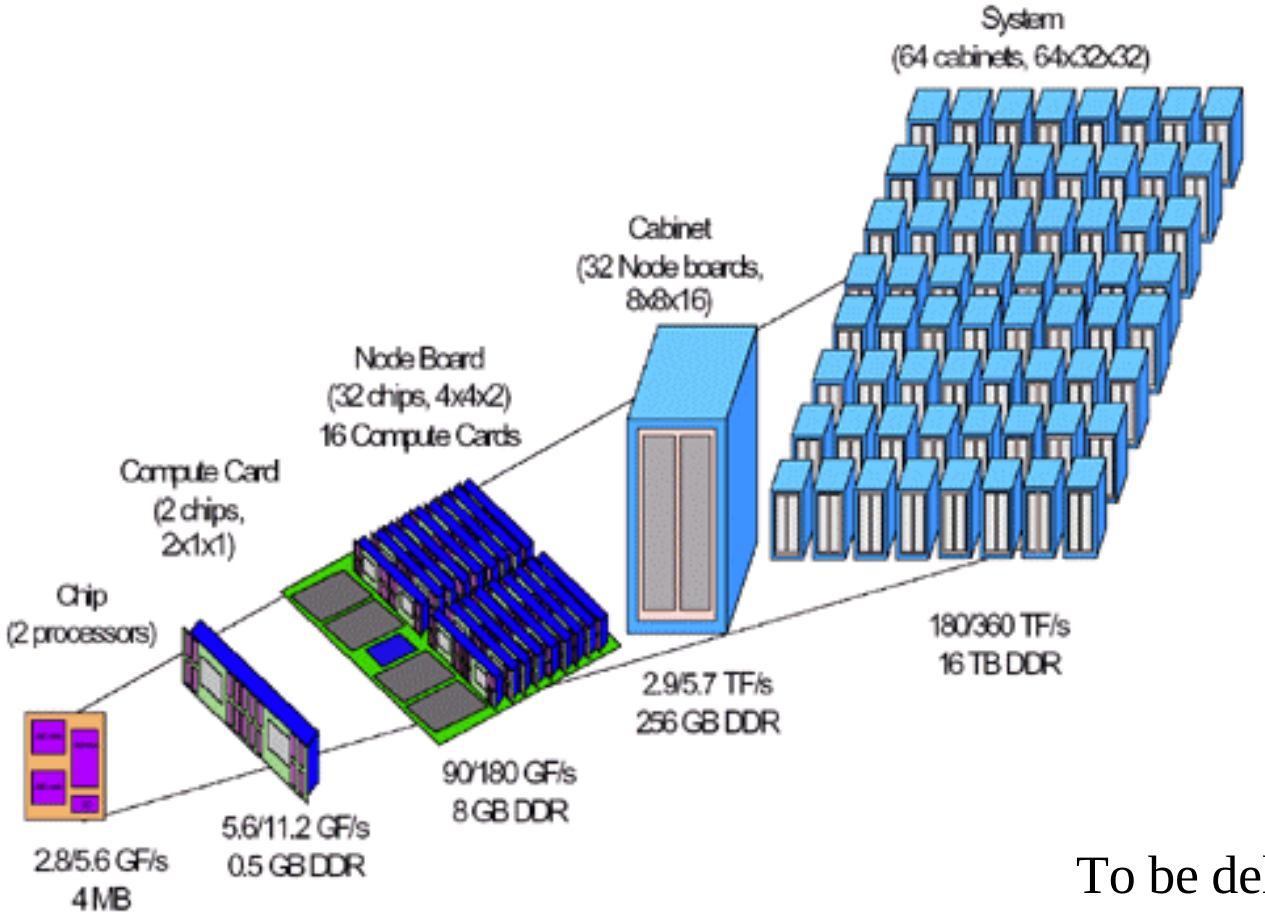
…and now for SciDAC

- IBM Power3+ SMP
- 16 procs per node
- 208 nodes
- 24 Gflop/s per node
- 5 Tflop/s (upgraded to 10, Feb 2003)

- IBM Power4 Regatta
- 32 procs per node
- 24 nodes
- 166 Gflop/s per node
- 4Tflop/s (10 in 2003)

New architecture on horizon: Blue Gene/L

- 180 Tflop/s configuration (65536 dual processor chips)
- Closely related to QCDOC prototype (IBM system-on a chip)
- Ordered for LLNL institutional computing (not ASCI)



To be delivered 2004

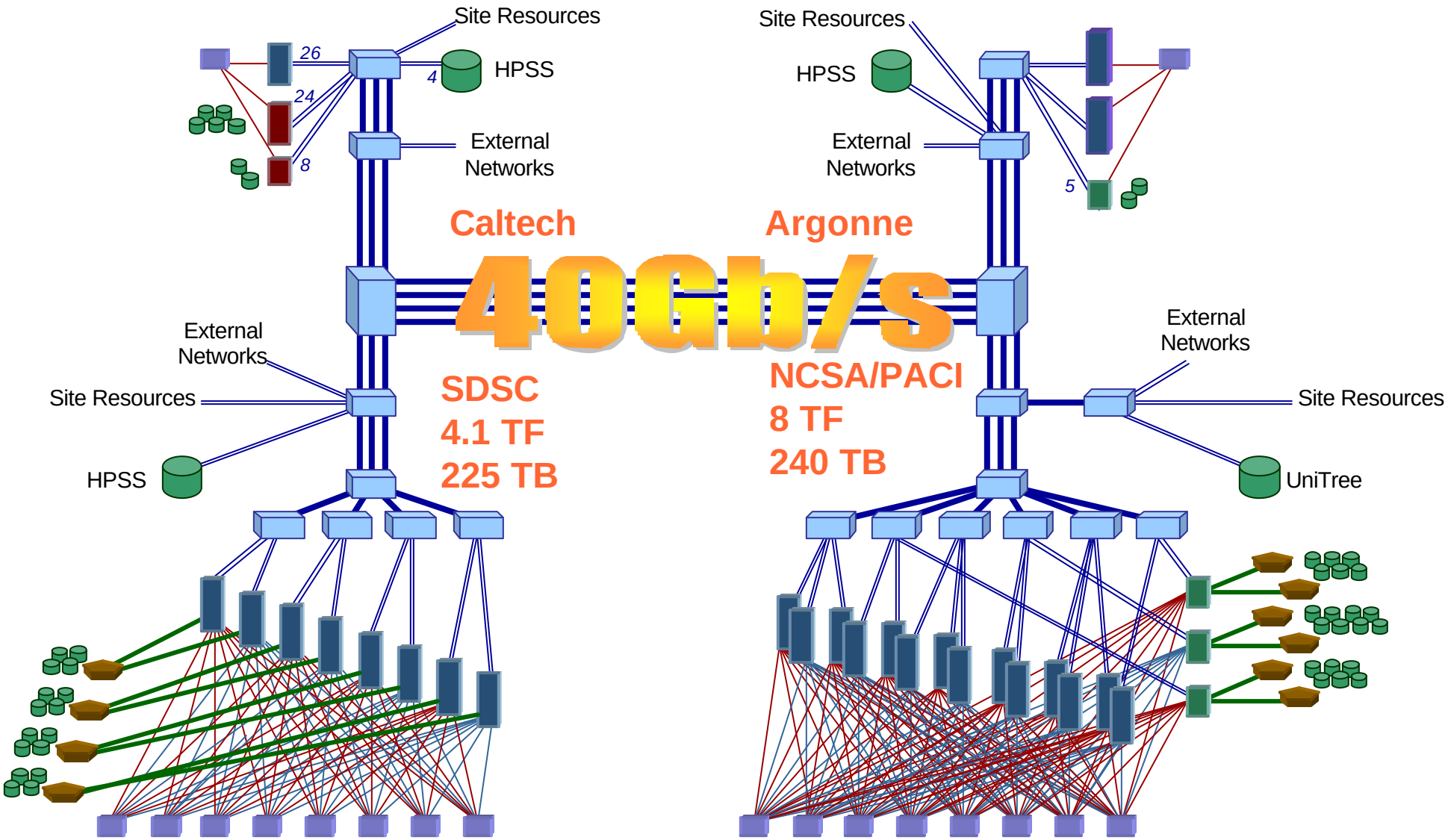
New architecture just arrived: Cray X1

- **Massively parallel-vector machine highly desired by global climate simulation community**
- **32-processor prototype ordered for evaluation**
- **Scale-up to 100 Tflop/s peak planned, if prototype proves successful**

Delivered to ORNL 18 March 2003

NSF's 13.6 TF TeraGrid coming on line

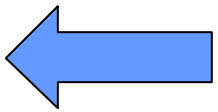
TeraGrid: NCSA, SDSC, Caltech, Argonne www.teragrid.org

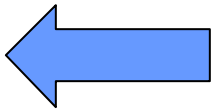


Does anyone recognize this sequence?

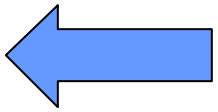
5120

8192

2304 

8192 

6656

1920 

2304

1540

3016

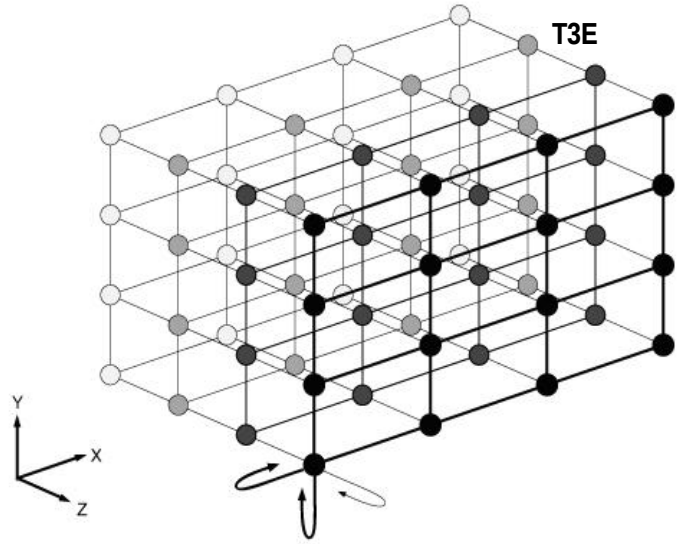
2560

Listed is the number of processors on the top 10 machines in the "Top500", compiled by the University of Mannheim, the University of Tennessee, and NERSC/LBNL: from the Japanese Earth Simulator (#1, 41 Tflop/s) to the French HP machine at CEA (#10, 5.1 Tflop/s). Machines #2, #3, and #6 are at Lawrence Livermore National Lab

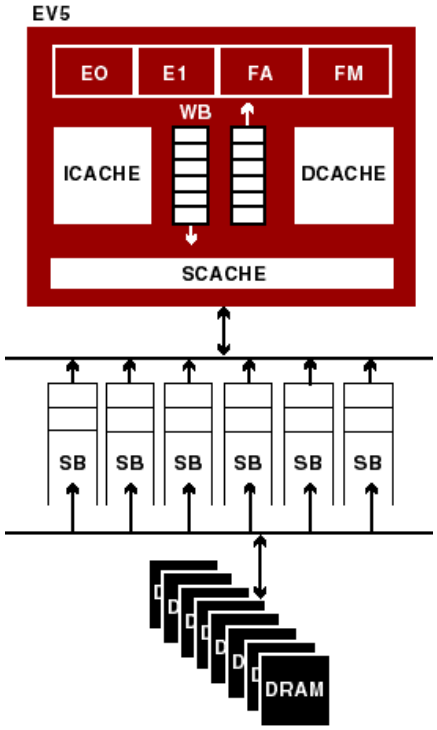
Algorithmic requirements from architecture

- **Must run on physically distributed memory units connected by message-passing network, each serving one or more processors with multiple levels of cache**

"horizontal" aspects



"vertical" aspects



Building platforms is the “easy” part

- Algorithms must be
  - highly concurrent and straightforwa[illegible]nce
  - latency tolera[illegible]
  - [illegible] temporal and s[illegible]cality)
  - highly scalable (in[illegible])
- Domain d[illegible]osition “natural” for all of these
- Domain decomposition also “natural” for software engineering
- Fortunate that mathematicians built up its theory in advance of requirements!

See "How scalable is DD in practice?" in DD-11 proceedings

Algorithmic requirement

- **Goal for algorithmic scalability: fill up memory of arbitrarily large machines to increase resolution, *while preserving nearly constant* running times* with respect to proportionally smaller problem on one processor**

*at worst logarithmically growing

Application properties

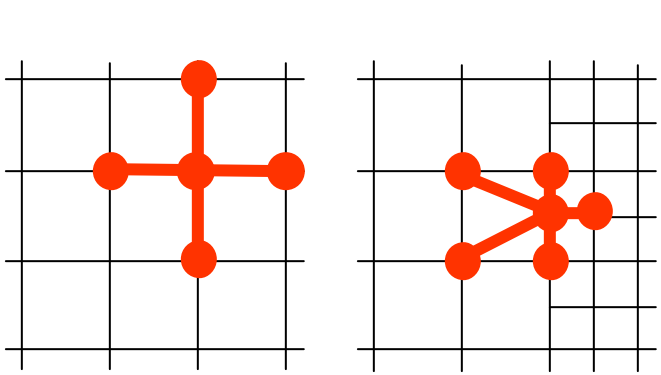
- After modeling and spatial discretization, we end up with large nonlinear algebraic system $F(u) = 0$ (which could come from $f(\dot{u}, u, t) = 0$, after implicit temporal discretization, at each time step)
- For PDEs, the Jacobian matrix $F'(u)$ is sparse
  - Each equation comes from a local flux balance
  - In unsteady case, timestep improves diagonal dominance
- For conservation law PDEs, there is a hierarchy of successively coarser approximate discretizations available (e.g., fusing control volumes)
- Discrete Green's function is generally global, with decaying tail

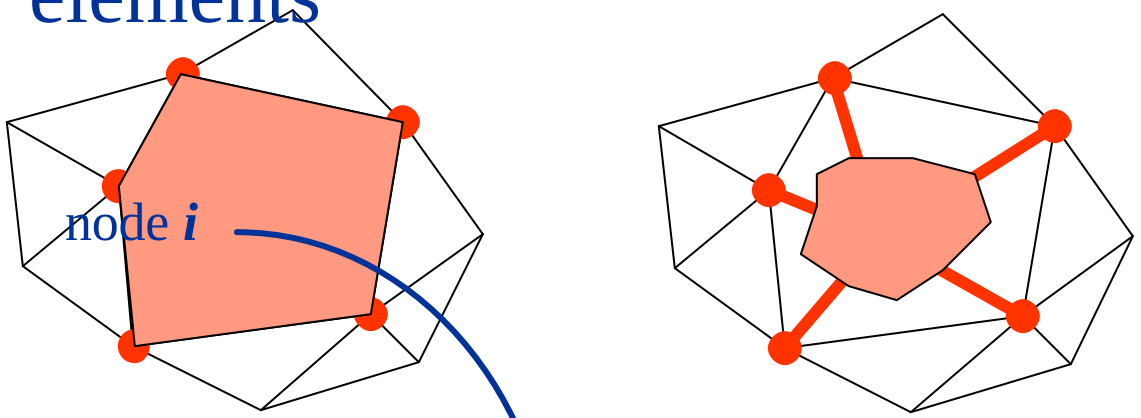
Dominant data structures are grid-based

finite differences

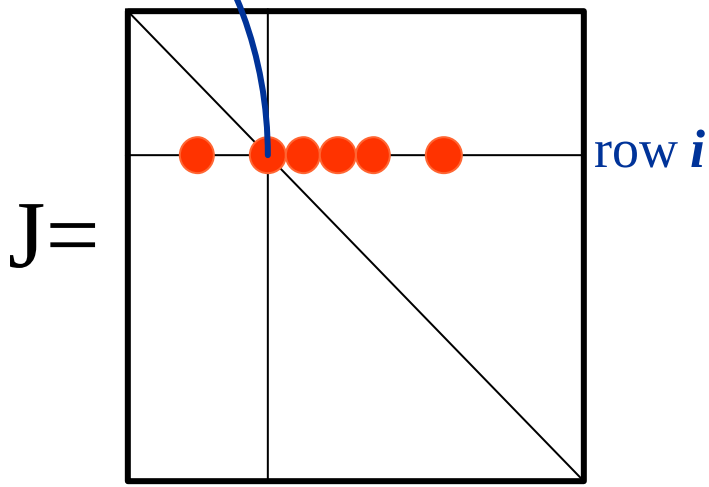
finite elements

finite volumes





All lead to problems with sparse Jacobian matrices



Decomposition strategies for $Lu=f$ in W

- **Operator decomposition**

$$L = \sum_k L_k$$

- **Function space decomposition**

$$f = \sum_k f_k \Phi_k, u = \sum_k u_k \Phi_k$$

- **Domain decomposition**

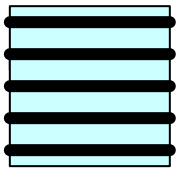
$$\Omega = \bigcup_k \Omega_k$$

Consider the implicitly discretized parabolic case

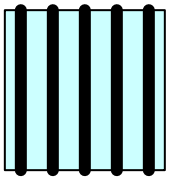
$$[\frac{I}{t} + L_x + L_y]u^{(k+1)} = \frac{I}{t}u^{(k)} + f$$

Operator decomposition

- **Consider ADI**



$$[\frac{I}{t/2}+L_x]u^{(k+1/2)}=[\frac{I}{t/2}-L_y]u^{(k)}+f$$



$$[\frac{I}{t/2}+L_y]u^{(k+1)}=[\frac{I}{t/2}-L_x]u^{(k+1/2)}+f$$

- **Iteration matrix consists of four multiplicative substeps per timestep**
  - **two sparse matrix-vector multiplies**
  - **two sets of unidirectional bandsolves**
- **Parallelism *within* each substep**
- **But global data exchanges *between* bandsolve substeps**

Function space decomposition

- **Consider a spectral Galerkin method**

$$u(x,y,t) = \sum_{j=1}^{N} a_j(t)\Phi_j(x,y)$$

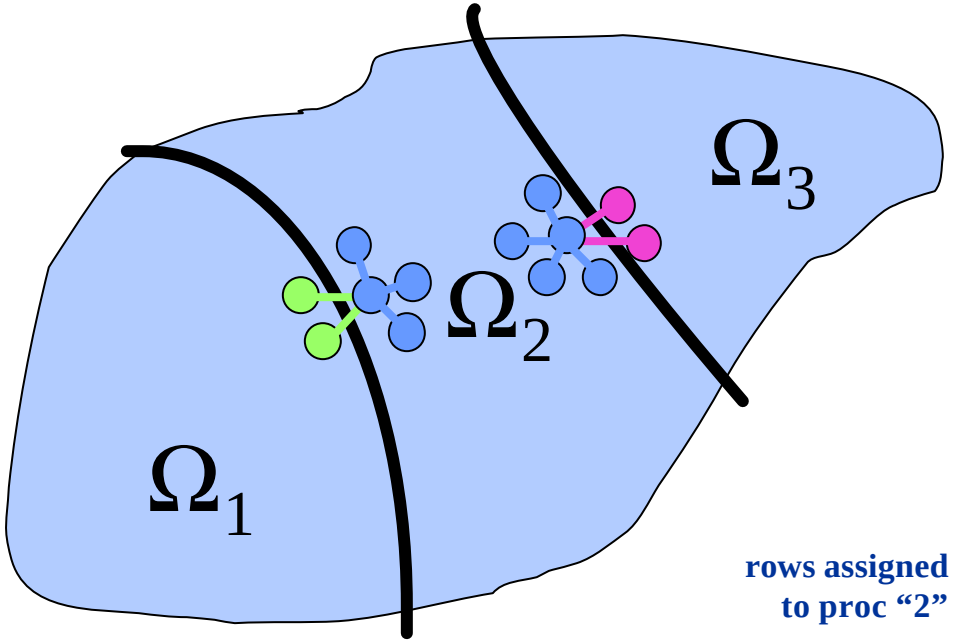
$$\frac{d}{dt}(\Phi_i, u) = (\Phi_i, \mathcal{L}u) + (\Phi_i, f), i = 1,\ldots,N$$

$$\sum_j (\Phi_i, u)\frac{da_j}{dt} = \sum_j (\Phi_i, \mathcal{L}\Phi_j)a_j + (\Phi_i, f), i = 1,\ldots,N$$

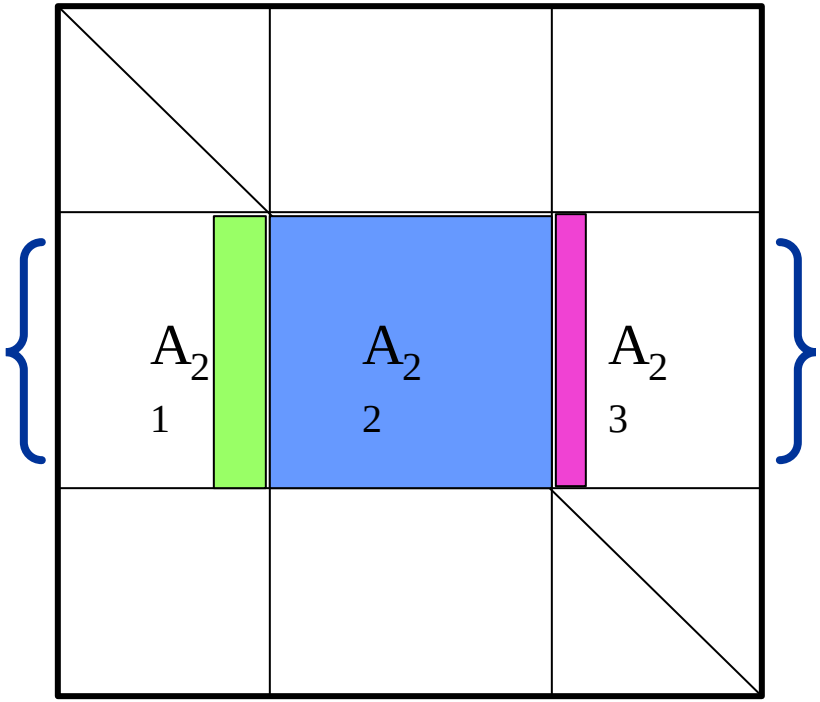
$$\frac{da}{dt} = M^{-1}Ka + M^{-1}f$$

- **System of ordinary differential equations**
- **Perhaps** $M \equiv [(\Phi_j, \Phi_i)], K \equiv [(\Phi_j, \mathcal{L}\Phi_i)]$ **are diagonal matrices**
- **Parallelism across spectral index**
- **But global data exchanges to *transform back* to physical variables at each step**

SPMD parallelism w/domain decomposition



rows assigned to proc "2"



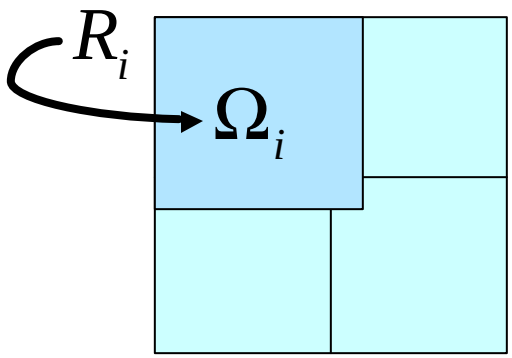
Partitioning of the grid induces block structure on the Jacobian

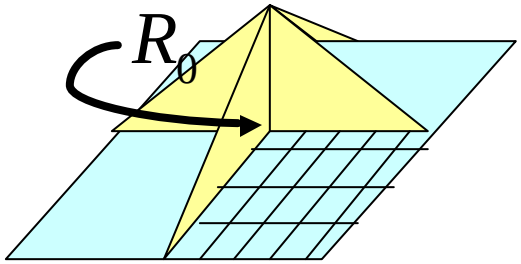
Schwarz domain decomposition method

- **Consider restriction and extension operators for subdomains,** R_i, R_i^T, **and for possible coarse grid,** R_0, R_0^T
- **Replace discretized** $Au = f$ **with**

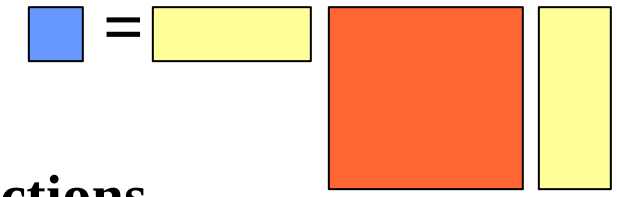
$$B^{-1}Au = B^{-1}f$$

$$B^{-1} = R_0^T A_0^{-1} R_0 + \sum_i R_i^T A_i^{-1} R_i$$





$$A_i = R_i A R_i^T$$



- **Solve by a Krylov method**
- **Matrix-vector multiplies with**
  - **parallelism on each subdomain**
  - **nearest-neighbor exchanges, global reductions**
  - **possible small global system (not needed for parabolic case)**

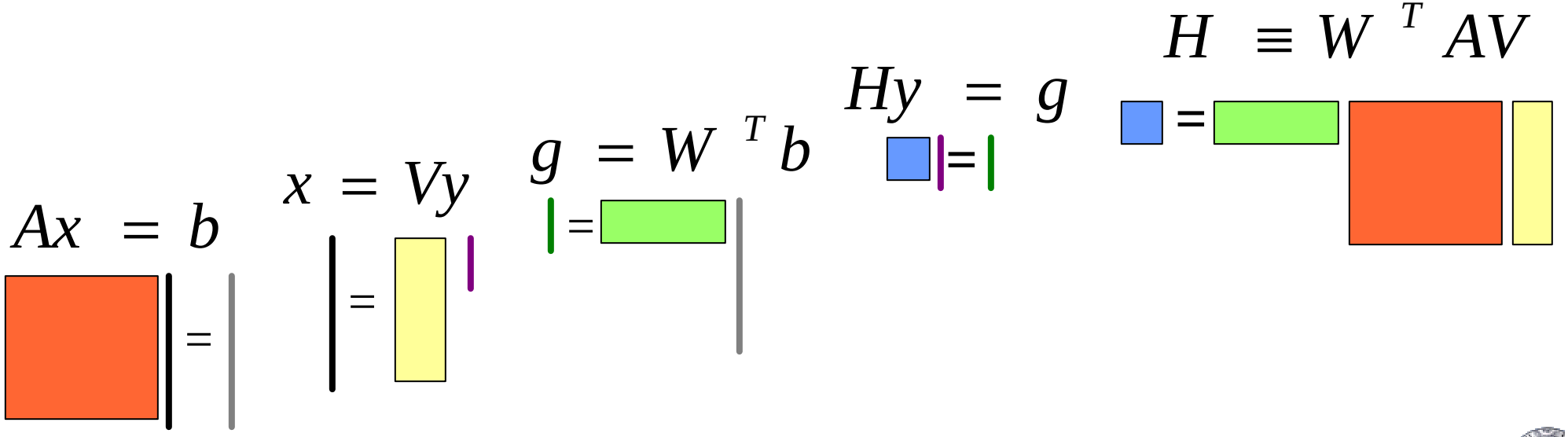
Recall Krylov methods

- Given $Ax = b, A \in \Re^{n \times n}$ and iterate x^0, we wish to generate a basis $V = \{v_1, v_2, ..., v_k\} \in \Re^{n \times k}$ for x ($x \approx Vy$) and a set of coefficients $\{y_1, y_2, ..., y_k\}$ such that x^k is a best fit in the sense that $y \in \Re^k$ minimizes $\| AVy - b \|$
- Krylov methods define a complementary basis $W = \{w_1, w_2, ..., w_k\} \in \Re^{n \times k}$ so that $W^T (AVy - b) = 0$ may be solved for ***y***
- In practice $k << n$ and the bases are grown from seed vector $r^0 = Ax^0 - b$ via recursive multiplication by A and Gram-Schmidt

Algebraic “picture” of Krylov iteration

- **Krylov iteration is an algebraic Galerkin method (or more generally Petrov-Galerkin method) for converting a high-dimensional linear system into a lower-dimensional linear system**
- **E.g., conjugate gradients (CG) for symmetric, positive definite systems, and generalized minimal residual (GMRES) for nonsymmetry or indefiniteness**

$Ax = b$ $x = Vy$ $g = W^T b$ $Hy = g$ $H \equiv W^T AV$



Now, let's compare!

- **Operator decomposition (ADI)**
  - **natural row-based assignment requires *global all-to-all, bulk* data exchanges in each step (for transpose)**
- **Function space decomposition (Fourier)**
  - **Natural mode-based assignment requires *global all-to-all, bulk* data exchanges in each step (for transform)**
- **Domain decomposition (Schwarz)**
  - **Natural domain-based assignment requires *local surface* data exchanges, *global reductions*, and *optional small global* problem**

(Of course, domain decomposition can be interpreted as a *special* operator or function space decomposition)

Estimating scalability of stencil computations

- **Given complexity estimates of the leading terms of:**
  - **the concurrent computation (per iteration phase)**
  - **the concurrent communication**
  - **the synchronization frequency**
- **And a bulk synchronous model of the architecture including:**
  - **internode communication (network topology and protocol reflecting horizontal memory structure)**
  - **on-node computation (effective performance parameters including vertical memory structure)**
- **One can estimate optimal concurrency and optimal execution time**
  - **on per-iteration basis, or overall (by taking into account any granularity-dependent convergence rate)**
  - **simply differentiate time estimate in terms of *(N,P)* with respect to *P*, equate to zero and solve for *P* in terms of *N***

Estimating 3D stencil costs (per iteration)

- grid points in each direction n, total work $N=O(n^3)$
- processors in each direction p, total procs $P=O(p^3)$
- memory per node requirements $O(N/P)$
- execution time per iteration $A\, n^3/p^3$
- grid points on side of each processor subdomain n/p
- neighbor communication per iteration $B\, n^2/p^2$
- cost of global reductions in each iteration $C \log p$ or $C\, p^{(1/d)}$
  - C includes synchronization frequency
- same dimensionless units for measuring A, B, C
  - e.g., cost of scalar floating point multiply-add

3D stencil computation illustration

Rich local network, tree-based global reductions

- **total wall-clock time per iteration**

$$T(n,p) = A\frac{n^3}{p^3} + B\frac{n^2}{p^2} + C\log p$$

- **for optimal p,** $\frac{\partial T}{\partial p} = 0$ **, or** $-3A\frac{n^3}{p^4} - 2B\frac{n^2}{p^3} + \frac{C}{p} = 0,$

or (with $q \equiv \frac{32B^3}{243A^2C}$ **),**

$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{q})\right]^{1/3} + \left[1-(1-\sqrt{q})\right]^{1/3}\right)\cdot n$$

- **without "speeddown," p can grow with n**
- **in the limit as** $B/C \to 0$

$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

3D stencil computation illustration

Rich local network, tree-based global reductions

- **optimal running time**

$$T(n, p_{opt}(n)) = \frac{A}{r^3} + \frac{B}{r^2} + C\log(rn),$$

where

$$r = \left(\frac{3A}{2C}\right)^{1/3} \left(\left[1 + (1 - \sqrt{q})\right]^{1/3} + \left[1 - (1 - \sqrt{q})\right]^{1/3} \right)$$

- **limit of infinite neighbor bandwidth, zero neighbor latency ($B \to 0$)**

$$T(n, p_{opt}(n)) = C\left[\log n + \frac{1}{3}\log\frac{A}{C} + const.\right]$$

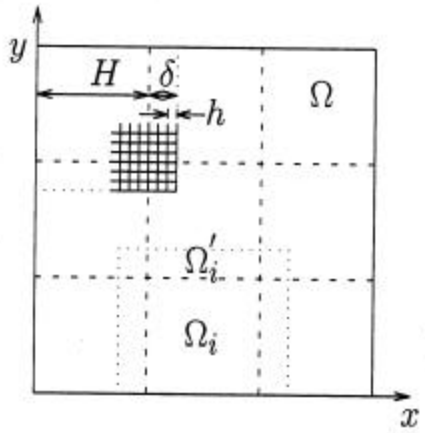
(This analysis is on a per iteration basis; fuller analysis would multiply this cost by an iteration count estimate that generally depends on *n* and *p*.)

Scalability results for DD stencil computations

- **With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales *linearly* with problem size**
- **With 3D torus-based global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales as *three-fourths* power of problem size (almost “scalable”)**
- **With common network bus (heavy contention):**
  - **optimal number of processors scales as *one-fourth* power of problem size (not “scalable”)**
  - **bad news for conventional Beowulf clusters, but see 2000 Bell Prize “price-performance awards”, for multiple NICs**

Factoring convergence into estimates

- **Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system**
- **In terms of *N* and *P*, where for *d*-dimensional isotropic problems, $N=h^{-d}$ and $P=H^{-d}$, for mesh parameter *h* and subdomain diameter *H*, iteration counts may be estimated as follows:**



Preconditioning Type	in 2D	in 3D
Point Jacobi	? ($N^{1/2}$)	? ($N^{1/3}$)
Domain Jacobi (d=0)	? ($(NP)^{1/4}$)	? ($(NP)^{1/6}$)
1-level Additive Schwarz	? ($P^{1/2}$)	? ($P^{1/3}$)
2-level Additive Schwarz	? (1)	? (1)

Where do these results come from?

- **Point Jacobi is well known (see any book on the numerical analysis of elliptic problems)**
- **Subdomain Jacobi has interesting history (see ahead a few slides)**
- **Schwarz theory is neatly and abstractly summarized in Section 5.2 of book by Smith, Bjorstad & Gropp ("Widlund School")**
  - **condition number,** $k = w\,[1+r(e)]\,C_0^2$
  - C_0^2 **is a splitting constant for the subspaces of the decomposition**
  - $r(e)$ **is a measure of the orthogonality of the subspaces**
  - w **is a measure of the approximation properties of the subspace solvers (can be unity for exact subdomain solves)**
  - **These properties are estimated for different subspaces, different operators, and different subspace solvers and the "crank" is turned**

Comments on the Schwarz results

- **Basic Schwarz estimates are for:**
  - ***self-adjoint*** **elliptic operators**
  - ***positive definite*** **operators**
  - ***exact*** **subdomain solves,** A_i^{-1}
  - ***two-way*** **overlapping with** R_i, R_i^T
  - ***generous*** **overlap,** $d=O(H)$ **(otherwise 2-level result is** $O(1+H/d)$**)**
- **Extensible to:**
  - ***nonself-adjointness*** **(e.g, convection)**
  - ***indefiniteness*** **(e.g., wave Helmholtz)**
  - ***inexact*** **subdomain solves**
  - ***one-way*** **overlap communication ("restricted additive Schwarz")**
  - ***small*** **overlap**

Comments on the Schwarz results, cont.

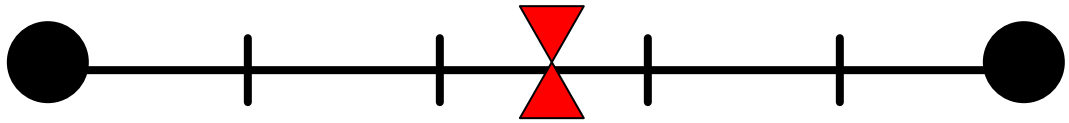
- **Theory still requires "sufficiently fine" coarse mesh**
  - **However, coarse space need *not* be nested in the fine space or in the decomposition into subdomains**
- **Practice is better than one has any right to expect**

> "In theory, theory and practice are the same ...
> In practice they're not!"
> — *Yogi Berra*

- **Wave Helmholtz (e.g., acoustics) is delicate at high frequency:**
  - **standard Schwarz Dirichlet boundary conditions can lead to undamped resonances within subdomains,** $u_\Gamma = 0$
  - **remedy involves Robin-type transmission boundary conditions on subdomain boundaries,** $(u + a\ \partial u / \partial n)_\Gamma = 0$

Block Jacobi preconditioning: 1D example

Consider the scaled F.D. Laplacian on an interval:

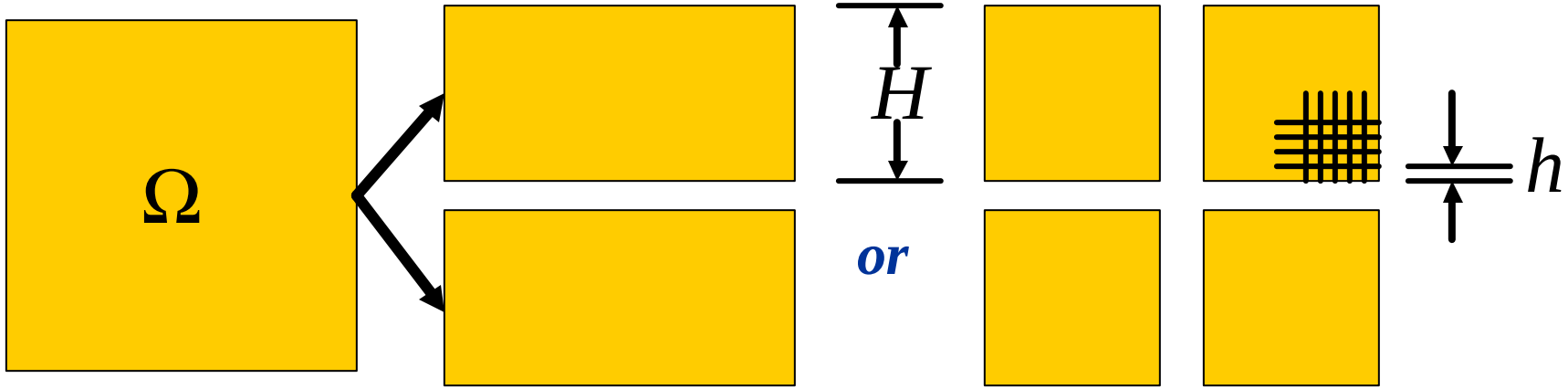


$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \bigcirc & \\ & \bigcirc & 2 & -1 \\ & & -1 & 2 \end{bmatrix}$$

$$B^{-1}A = I - \begin{bmatrix} & & \tfrac{1}{3} & \\ & & \tfrac{2}{3} & \\ & \tfrac{2}{3} & & \\ & \tfrac{1}{3} & & \end{bmatrix}$$

Bound on block Jacobi preconditioning

- **Consider decomposition of 1D, 2D, or 3D domain into subdomains by cutting planes**



- **Well known (e.g., late 1980's TR by Dryja & Widlund) that zero overlap Schwarz on elliptic problem improves conditioning from $O(h^{-2})$ for native problem to $O(H^{-1}h^{-1})$**

Mirror result from linear algebra

- Chang & Schultz (1994) proved same result from algebraic approach, from eigenanalysis of $(B^{-1}A)$, where A is F.D. Laplacian in 1D, 2D, or 3D, and B is A with entries removed by arbitrary cutting planes
- Their **Theorem 2.4.7**: Given $n \times n \times n$ grid, cut by
  - q planes in x (slabs)
  - q planes in x or y (beams)
  - q planes in $x, y,$ or z (subcubes)

  (with cuts anywhere) then $k(B^{-1}A) \le qn + q + 1$
- Note: $q = O(H^{-1})$ and $n = O(h^{-1})$ if cut evenly
- Proof: eigenanalysis of low-rank matrices $I - (B^{-1}A)$

Mirror results from graph theory

- **Boman & Hendrickson (2003) proved same result from graph-theoretic approach, using their new "support theory" (SIMAX, to appear)**
- **Section 9 of SIMAX paper "Support Theory for Preconditioning," using *congestion-dilation* lemma from graph theory (Vaidya *et al.*) derives $O(h^{-2})$, for *point* Jacobi**
- **Extended January 2003 to *block* Jacobi, to get $O(H^{-1}h^{-1})$**
- **Fascinating to see how many different tools can be used for this divide and conquer preconditioning idea**

"Unreasonable effectiveness" of Schwarz

- **When does the sum of partial inverses equal the inverse of the sums? When the decomposition is right!**

  Let $\{r_i\}$ be a complete set of orthonormal row eigenvectors for A : $r_i A = a_i r_i$ **or** $a_i = r_i A r_i^T$

  Then

$$A = \Sigma_i r_i^T a_i r_i$$

  and

$$A^{-1} = \Sigma_i r_i^T a_i^{-1} r_i = \Sigma_i r_i^T (r_i A r_i^T)^{-1} r$$

  — the Schwarz formula!

- **Good decompositions are a compromise between conditioning and parallel complexity, in practice**

Basic Domain Decomposition Concepts

- Iterative correction
- Schwarz preconditioning
- Schur preconditioning

"Advanced" Domain Decomposition Concepts

- Polynomial combinations of Schwarz projections
- Schwarz-Schur combinations
  - Neumann-Neumann/FETI (Schwarz on Schur)
  - LNKS (Schwarz inside Schur) (Friday afternoon)
- Nonlinear Schwarz (Thursday afternoon)

Iterative correction

- **The most basic idea in iterative methods**

$$u \leftarrow u + B^{-1}(f - Au)$$

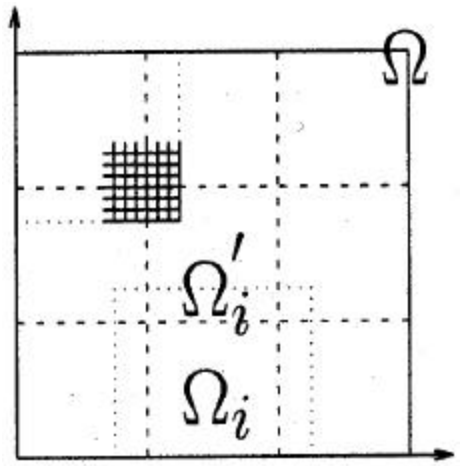
- **Evaluate residual accurately, but solve approximately, where B^{-1} is an approximate inverse to A**
- **A sequence of complementary solves can be used, e.g., with B_1 and B_2 one has**

$$u \leftarrow u + [B_1^{-1} + B_2^{-1} - B_2^{-1} A B_1^{-1}](f - Au)$$

- **Scale recurrence, e.g., with $B_2^{-1} = R^T (RAR^T)^{-1} R$, leads to *multilevel methods***
- **Optimal polynomials of $(B^{-1}A)$ leads to various *preconditioned Krylov methods***

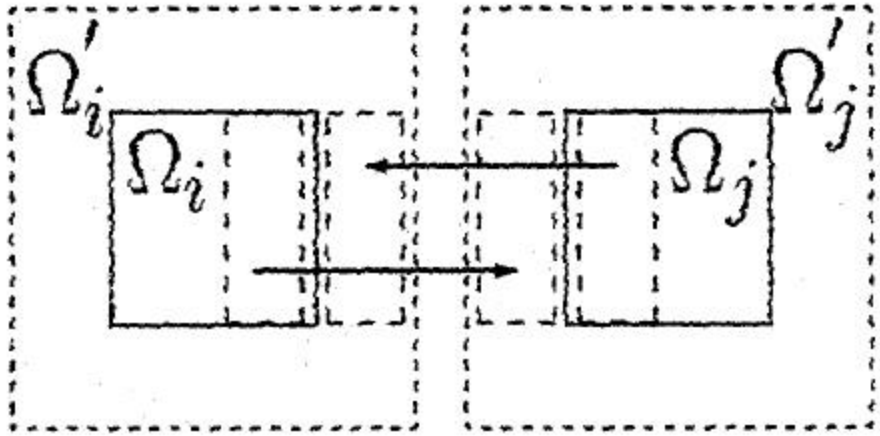
Schwarz Preconditioning

- Given $A x = b$, partition x into subvectors, corresp. to subdomains Ω_i of the domain Ω of the PDE, nonempty, possibly overlapping, whose union is all of the elements of $x \in \Re^n$



- Let Boolean rectangular matrix R_i extract the i^{th} subset of x :

$$x_i = R_i x$$



- Let $A_i = R_i A R_i^T$

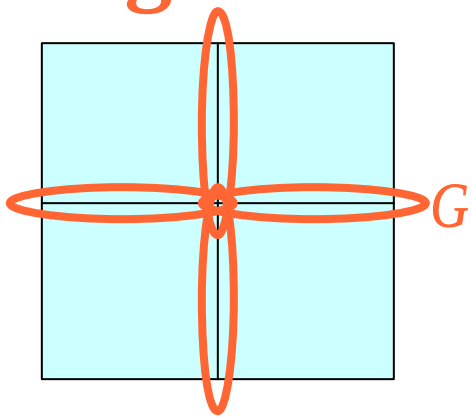
$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i$$

The Boolean matrices are gather/scatter operators, mapping between a global vector and its subdomain support

Schur complement substructuring

- **Given a partition**

$$\begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$$



- **Condense:**

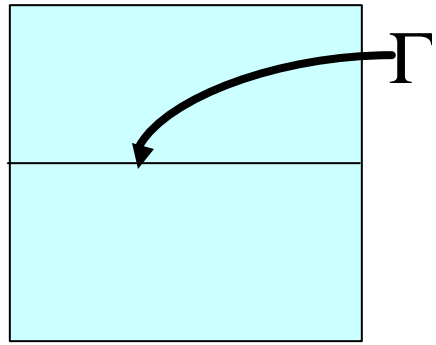
$$Su_\Gamma = g \qquad S \equiv A_{\Gamma\Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} \qquad g \equiv f_\Gamma - A_{\Gamma i} A_{ii}^{-1} f_i$$

- **Properties of the Schur complement:**
  - **smaller than original A, but generally dense**
  - **expensive to form, to store, to factor, and to solve**
  - **better conditioned than original A**
- **Therefore, solve iteratively, with action of S on each Krylov vector**

Schur preconditioning

- Let M^{-1} be a good preconditioner for S
- Let

$$B^{-1} = \left(\begin{bmatrix} A_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix} \begin{bmatrix} I & A_{ii}^{-1} A_{i\Gamma} \\ 0 & M \end{bmatrix} \right)^{-1}$$



- Then B^{-1} is a preconditioner for A
- So, instead of $M^{-1} S u_\Gamma = M^{-1} g$, use full system

$$B^{-1} \begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = B^{-1} \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$$

- Here, solves with A_{ii} may be done approximately since all degrees of freedom are retained

Schwarz polynomials

- **Polynomials of Schwarz projections that are combinations of additive and multiplicative may be appropriate for certain implementations**
- **We may solve the fine subdomains concurrently and follow with a coarse grid (redundantly/cooperatively)**

$$u \leftarrow u + \Sigma_i B_i^{-1}(f - Au)$$

$$u \leftarrow u + B_0^{-1}(f - Au)$$

- **This leads to algorithm "Hybrid II" in S-B-G'96:**

$$B^{-1} = B_0^{-1} + (I - B_0^{-1}A)(\Sigma_i B_i^{-1})$$

- **Convenient for "SPMD" (single prog/multiple data)**

Schwarz-on-Schur

- **Preconditioning the Schur complement is complex in and of itself; Schwarz is used on the reduced problem**
- **Neumann-Neumann**

$$M^{-1} = \Sigma_i D_i R_i^T S_i^{-1} R_i D_i$$

- **Balancing Neumann-Neumann**

$$M^{-1} = M_0^{-1} + (I - M_0^{-1} S)(\Sigma_i D_i R_i^T S_i^{-1} R_i D_i)(I - S M_0^{-1})$$

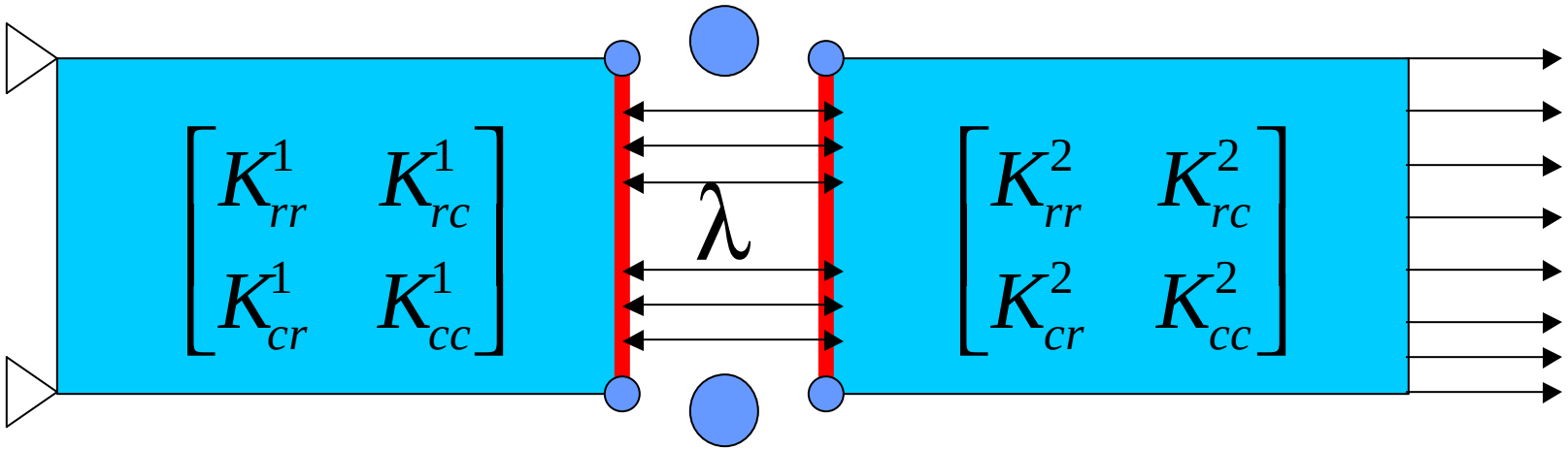
- **Other variants:**
  - **Bramble-Pasciak-Schatz**
  - **multigrid on the Schur complement**

FETI overview (next 8 slides, c/o K. Pierson)

- **Finite Element Tearing and Interconnection**
- **Domain decomposition iterative linear solver that uses Lagrange multipliers to solve $A\,x = b$**
- **Numerically scalable (convergence rate bounded)**
- **Parallel scalability up to 1000s of processors**
- **Used to solve large scale finite element models (10-100 million equations)**
- **Sandia's *Salinas* and *SIERRA* implementations of the Dual/Primal FETI method (FETI-DP) won Gordon Bell Prize in "special" category in 2002 (w/ C. Farhat, invited to speak Monday afternoon at DD-15)**

Interface Continuity

- Subscript *r* for interior DOFs; subscript *c* for interface DOFs
- Interface continuity enforced through Lagrange multipliers



At Convergence: $\sum_{s=1}^{N_s} B_r^s u_r^s = 0$

Equilibrium & Compatibility

- The ***u***'s are primal variables; the ***?*** are dual variables
- FETI-DP master system (symmetric) is:

$$\begin{bmatrix} K_{rr}^{1} & \cdots & 0 & K_{rc}^{1}B_{c}^{1} & B_{r}^{1^T} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & K_{rr}^{N_s} & K_{rc}^{N_s}B_{c}^{N_s} & B_{r}^{N_s{}^T} \\ B_{c}^{1^T}K_{cr}^{1} & \cdots & B_{c}^{N_s{}^T}K_{cr}^{N_s} & \sum_{s=1}^{N_s} B_{c}^{s^T}K_{cc}^{s}B_{c}^{s} & 0 \\ B_{r}^{1} & \cdots & B_{r}^{N_s} & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{r}^{1} \\ \vdots \\ u_{r}^{N_s} \\ u_{c} \\ l \end{bmatrix} = \begin{bmatrix} f_{r}^{1} \\ \vdots \\ f_{r}^{N_s} \\ \sum_{s=1}^{N_s} B_{c}^{s^T} f_{c} \\ 0 \end{bmatrix}$$

FETI-DP interface problem

After some algebraic manipulation …

$$(F_{rr} + F_{rc} K_{cc}^{*-1} F_{rc}^{T}) l = d_r - F_{rc} K_{cc}^{*-1} f_c^*$$

$$F_{rr} = \sum_{s=1}^{N_s} B_r^s K_{rr}^{s-1} B_r^{sT} \qquad F_{rc} = \sum_{s=1}^{N_s} B_r^s K_{rr}^{s-1} K_{rc}^s B_c^s$$

$$d_r = \sum_{s=1}^{N_s} B_r^s K_{rr}^{s-1} f_r^s \qquad f_c^* = \sum_{s=1}^{N_s} B_c^{sT} (f_c^s - K_{rc}^{sT} K_{rr}^{s-1} f_r^s$$

FETI-DP coarse problem

$$K_{cc}^{*} = \sum_{s=1}^{N_s} B_c^{s^T} \left(K_{cc}^{s} - K_{cr}^{s} K_{rr}^{s^{-1}} K_{rc}^{s} \right) B_c^{s}$$

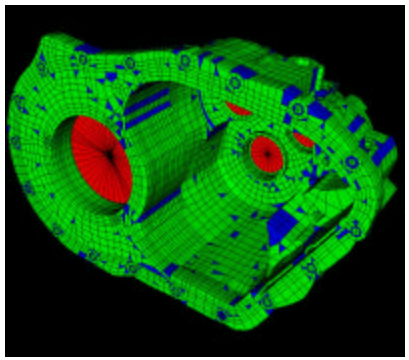
- Couples all subdomains
- Global propagation of residual error
- Solved with parallel distributed sparse solver

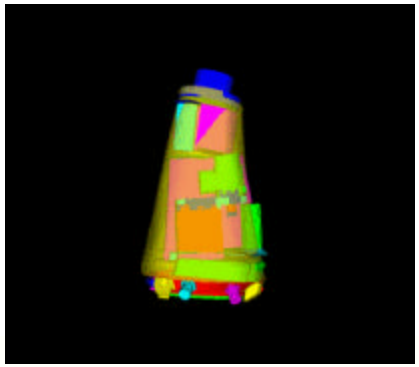
Solving $A\,x = b$ using FETI-DP

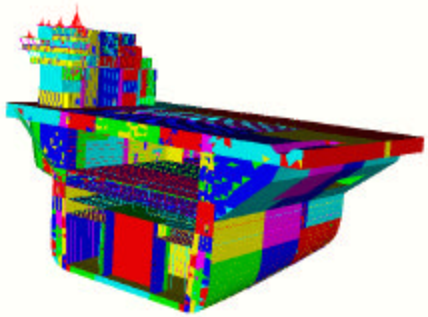
- Construct & factorize subdomain operators
- Construct & factorize preconditioner
- Construct & factorize coarse grid
- Solve for Lagrange multipliers (with preconditioned Conjugate Gradients)
- Solve for displacements using back substitution

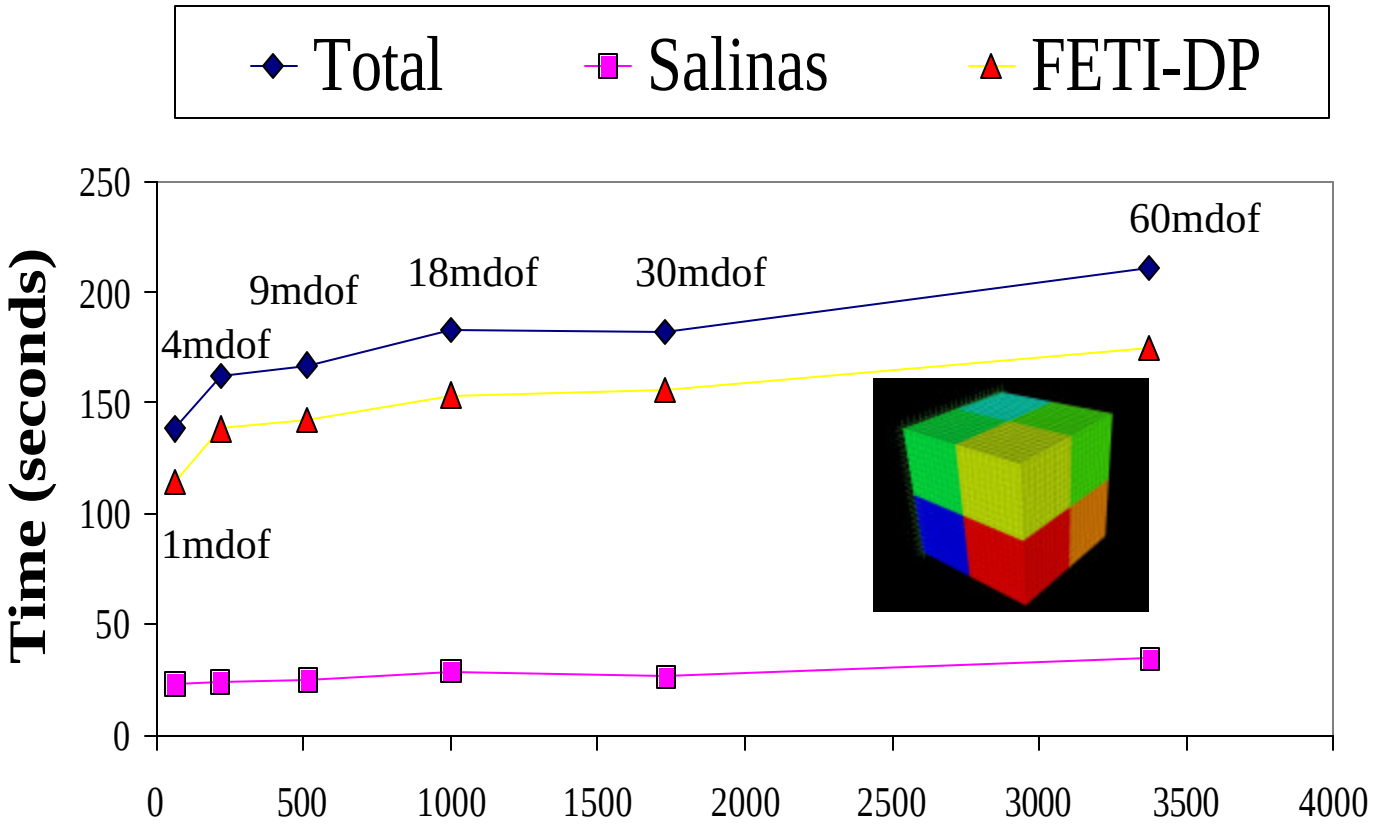
FETI-DP for structural mechanics

- **Numerically scalable, hardware scalable solutions for realistic solid/shell models**
- **Used in Sandia applications Salinas, Adagio, Andante**









ASCI-White Processors

c/o C. Farhat and K. Pierson

Sandia National Laboratories

Agenda for future DD research

- High concurrency (100,000 processors)
- Asynchrony
- Fault tolerance
- Automated tuning of algorithm (to application and to architecture)
- Integration of “forward” simulation with studies of sensitivity, stability, and optimization

High Concurrency

Today

- 10,000 processors in a single room with tightly coupled network
- DD computations scale well, when provided with
  - network rich enough for parallel near neighbor communication
  - fast global reductions (complexity sublinear in processor count)

Future

- 100,000 processors, in a room or as part of a grid
- Most phases of DD computations scale well
  - favorable surface-to-volume comm-to-comp ratio
- However, latencies will nix frequent exact reductions
- Paradigm: *extrapolate* data in retarded messages; *correct (if necessary)* when message arrives, such as in *C(p,q,j)* schemes by Garbey and Tromeur-Dervout

Asynchrony

Today

- *A priori* partitionings for quasi-static meshes provide load-balanced computational tasks between frequent synchronization points
- Good load balance is critical to parallel scalability on 1,000 processors and more

Future

- Adaptivity requirements and far-flung, nondedicated networks will lead to idleness and imbalance at synchronization points
- Need algorithms with looser outer loops than global Newton-Krylov
- Can we design algorithms that are robust with respect to incomplete convergence of inner tasks, like inexact Newton?
- Paradigm: *nonlinear Schwarz* with regional (not global) nonlinear solvers where most execution time is spent

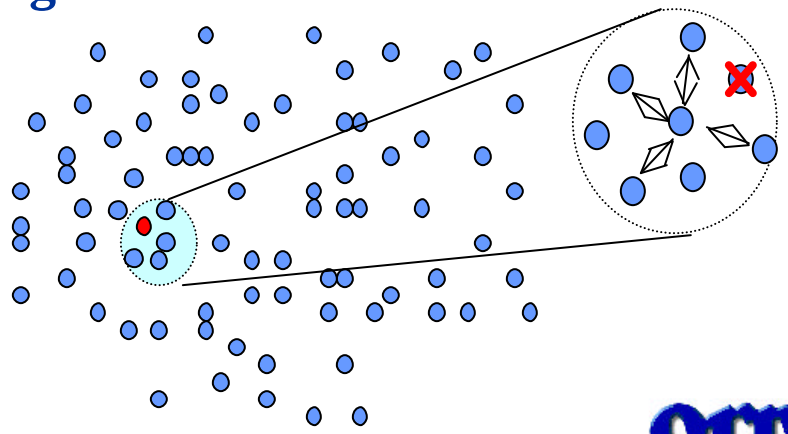
Fault Tolerance

Today

- Fault tolerance is not a driver in most scientific application code projects
- FT handled as follows:
  - Detection of wrong
    - System – in hardware
    - Framework – by runtime env
    - Library – in math or comm lib
  - Notification of application
    - Interrupt – signal sent to job
    - Error code returned by app process
  - Recovery
    - Restart from checkpoint
    - Migration of task to new hardware
    - Reassignment of work to remaining tasks

Future

- With 100,000 processors or worldwide networks, MTBF will be in minutes
- Checkpoint-restart could take longer than the time to next failure
- Paradigm: *naturally fault tolerant algorithms*, robust with respect to failure, such as a new FD algorithm at ORNL



c/o A. Geist

Automated Tuning

Today

- Knowledgeable user-developers parameterize their solvers with experience and theoretically informed intuition for:
  - problem size/processor ratio
  - outer solver type
  - Krylov solver type
  - DD preconditioner type
  - maximum subspace dimensions
  - overlaps
  - fill levels
  - inner tolerances
  - potentially many others

Future

- Less knowledgeable users required to employ parallel iterative solvers in taxing applications
- Need safe defaults and automated tuning strategies
- Paradigm: *parallel direct search (PDS) derivative-free optimization methods*, or other *machine learning* (ML), using overall parallel computational complexity as objective function and algorithm tuning parameters as design variables, to tune solver in preproduction trial executions

Integrated Software

Today

- Each analysis is a “special effort”
- Optimization, sensitivity analysis (e.g., for uncertainty quantification), and stability analysis to fully exploit and contextualize scientific results are *rare*

Future

- Analysis increasingly an “inner loop” around which more sophisticated science-driven tasks are wrapped
- Need PDE task functionality (e.g., residual evaluation, Jacobian evaluation, Jacobian inverse) exposed to optimization/sensitivity/stability algorithms
- Paradigm: integrated software based on common distributed data structures

Architecturally driven ideas in DD

- **Chaotic Relaxation (1969)**
- **Schwarz Waveform Relaxation (1997)**
- **Restricted Additive Schwarz (1997)**
- ***C(p,q,j)* schemes (2000)**
- **Hybrid MPI/OpenMP-based domain decomposition (2000)**

Chaotic Relaxation

- By Chazan & Miranker (1969)
- Basic idea: assign subsets of interdependent equations to different processors and relax concurrently, importing refreshed data on which a given processor depends “as available”
- Convergence (for certain problem classes) as long as no subset goes infinitely long without being updated
- Weak results from theory, but occasional encouraging numerical experiments, including Giraud (2001), who showed that chaotic relaxation can be marginally faster, both in execution time (from relaxation of synchrony) and in terms of actual floating point work done!

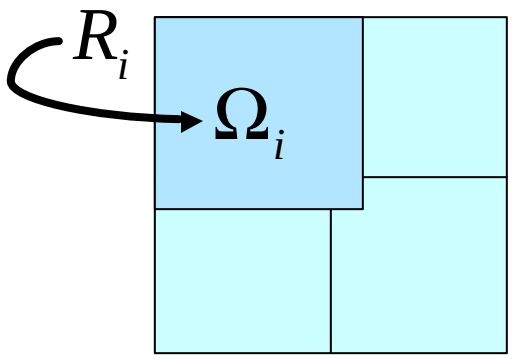
Schwarz Waveform Relaxation

- By Dauod & Gander (1997; see also DD-13 proceedings)
- Rather than exchanging messages at every time step in a space-time cylindrical domain, $(W,(0,T))$, over which a PDE is to be solved, solve in each domain over all time, and exchange interface data over $(0,T)$ at all overlapping Schwarz interfaces less frequently
- Nice convergence theory for parabolic problems using maximum principle
- Interesting for high-latency systems; also for multiphysics systems, since some subdomains can "step over" most restrictive time step arising in other domain
- Disadvantage: memory!

Schwarz Waveform Relaxation

- By Dauod & Gander (1997; see also DD-13 proceedings)
- Rather than exchanging messages at every time step in a space-time cylindrical domain, $(W,(0,T))$, over which a PDE is to be solved, solve in each domain over all time, and exchange interface data over $(0,T)$ at all overlapping Schwarz interfaces less frequently
- Nice convergence theory for parabolic problems using maximum principle
- Interesting for high-latency systems; also for multiphysics systems, since some subdomains can "step over" most restrictive time step arising in other domain
- Disadvantage: memory!

Restrictive Additive Schwarz



- By Cai & Sarkis (1997)
- Consider restriction and extension operators for subdomains, R_i, R_i^T
- Restrict either the restriction or the extension operator to ignore the overlap: $B^{-1} = \sum_i \overline{R}_i^T A_i^{-1} R_i$
- Solve as usual Krylov-Schwarz
- Saves 50% of communication, and actually converges faster in many cases; default in PETSc
- Active area in DD-13:
  - Cai, Dryja & Sarkis' RASHO shows that symmetry can be preserved if one projects to stay in a certain subspace
  - Frommer, Nabben & Szyld give an algebraic theory, including multiplicative RAS

C(p,q,j) schemes

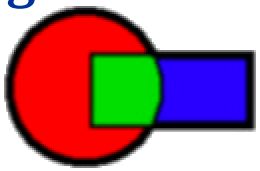
- By Garbey & Tromeur-Dervout (2000)
- To conquer high-latency environments, extrapolate missing boundary data (treating higher and lower Fourier modes differently), and to accommodate low bandwidth environments, reuse extrapolations over several steps
- Employ a posteriori checks against real boundary data when it appears, and adjust as necessary
- Nice results for parabolic problems in the "computational grid" environment

OpenMP/MPI tradeoffs

- By I. Charpentier & AHPIK software team (2000); explored for just two procs by Keyes *et al.* (1999)
- For p processors, rather than using p subdomains, use fewer, larger subdomains, and split a subdomain over several processors, using multithreaded subdomain solver, in a hybrid SPMD programming model
- Advantage: fewer subdomains, larger H, gives logarithmic or fractional power improvement in convergence for most DD methods (less information lost on subdomain cuts)

Conclusions/summary

- **Domain decomposition is the dominant paradigm in contemporary terascale PDE simulation**
- **Several freely available software toolkits exist, and successfully scale to thousands of tightly coupled processors for problems on quasi-static meshes**
- **Concerted efforts underway to make elements of these toolkits interoperate, and to allow expression of the best methods, which tend to be modular, hierarchical, recursive, and above all — *adaptive*!**
- **Many challenges loom at the "next scale" of computation**
- **Implementation of domain decomposition methods on parallel computers has inspired many useful variants of domain decomposition methods**
- **The past few years have produced an incredible variety of interesting results (in both the continuous and the discrete senses) in domain decomposition methods, with no slackening in sight**
- **Undoubtedly, new theory/algorithms will be part of the solution!**



EOF