Argonne NATIONAL LABORATORY

Recent PETSc Functionality in Linear Operators and Solvers

Hong Zhang and PETSc Team
Mathematics and Computer Science Div.
Argonne National Laboratory

SIAM Conference on Applied Linear Algebra
Valencia Spain, June 21, 2012

U.S. DEPARTMENT OF ENERGY

PETSc linear operators and solvers for extreme-scale computing

- **Sparse Matrix-Matrix Operations**
  - Computational kernels
    - Matrix product would be denser -

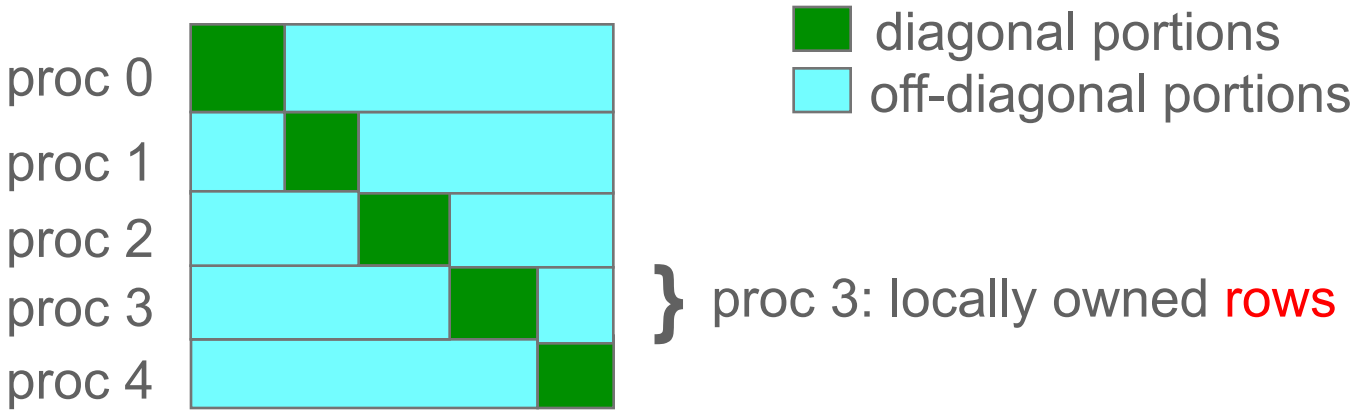
      avoid using these operations if possible
  - GAMG preconditioner requires these operations to be scalable to 10k+ processor cores
- **Krylov Solvers**
  - Fundamental sparse linear solvers
  - Global inner product (via MPI_Allreduce) is a major bottleneck for extreme-scale computing

Sparse Matrix-Matrix Operations (PETSc-3.3)

- C=AB (MatMatMult):
  - New parallel implementations, '-matmatmult_scalable'
- C=A^TB (MatTransposeMatMult):
  - sequential and parallel
- C=AB^T (MatMatTransposeMult):
  - sequential, '-matmattransmult_color'
- C=RAR^T (MatRARt):
  - sequential
- C=P^TAP (MatPtAP):
  - '-matptap_sparseaxpy 2' : slower, but uses a tmp array of local size (ptap->rmax) instead of global P->cmap->N.
- AB and P^TAP run smoothly on 13,824 cores in GAMG

PETSc Parallel Sparse Matrix (MATMPIAIJ)

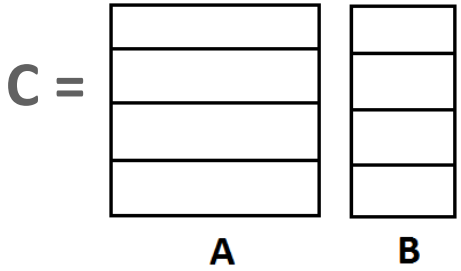
- Each process locally owns a submatrix of contiguously numbered global rows.
- Each submatrix consists of **diagonal** and **off-diagonal** parts.



Sparse Matrix-Matrix Operations (seqential)

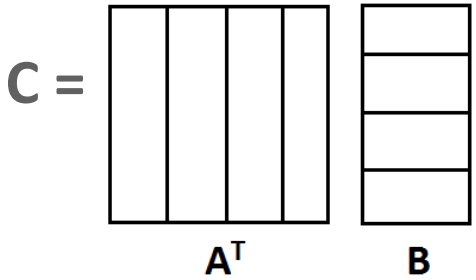
- **C=AB**

  O(Anz*Bnzi)



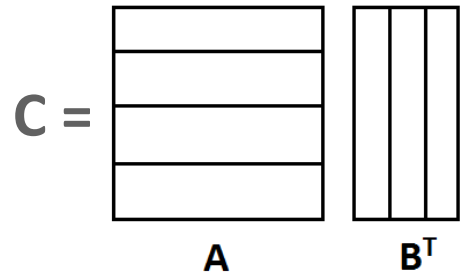
- **C=A^{T}B**

  O(Anz*Bnzi)



- **C=AB^T**

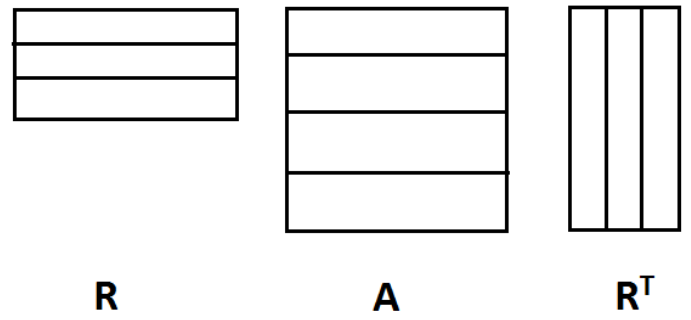
  O(Cnz *(Anzi + Bnzi))



Sparse Matrix-Matrix Operations (seq)

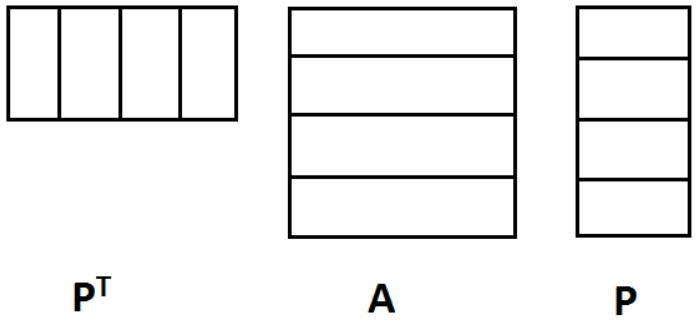
- **$C=RAR^T$**

  O(2Rnz*Anzi + Rm*Anzi*Anzi)



- **$C=P^TAP$**

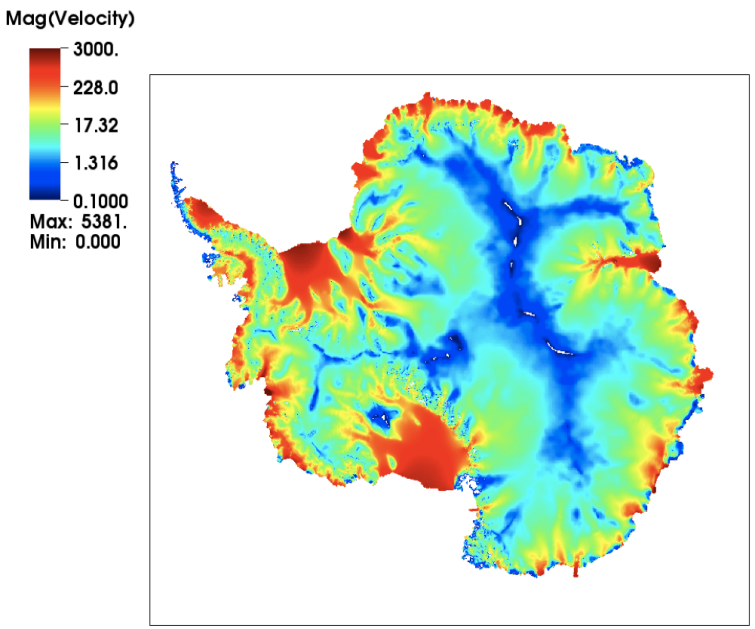
  O(2Anz*Pnzi)

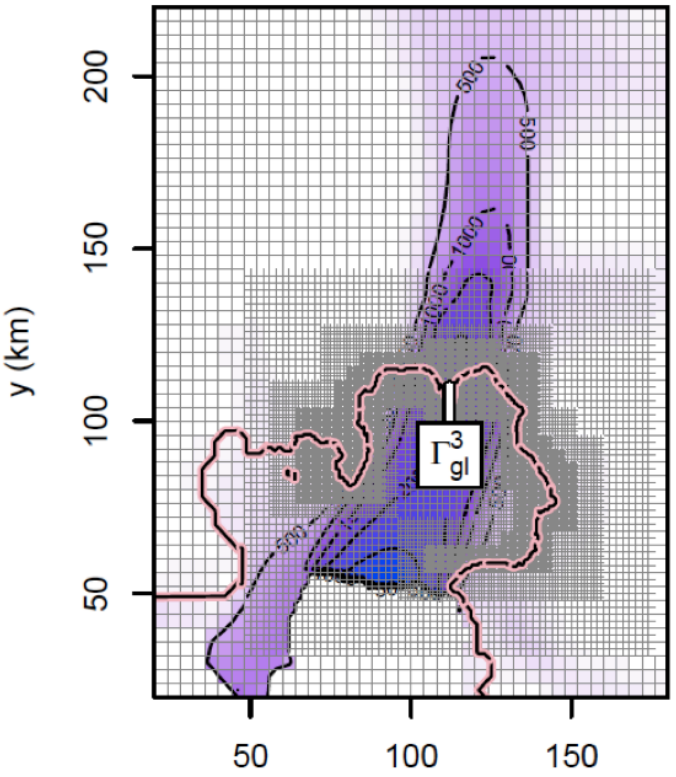


Sparse outer-product is preferable to sparse inner-product

GAMG (geometric algebraic multigrid)

- Unstructured Geometric-Algebraic Multigrid (GAMG)
- Complex geometries demand algebraic multigrid (AMG)
- Common parallel primitives for AMG:
  - $A_{i+1} = P^T A_i P$
  - $P = (I - \omega D^{-1} A) P_0$
  - …



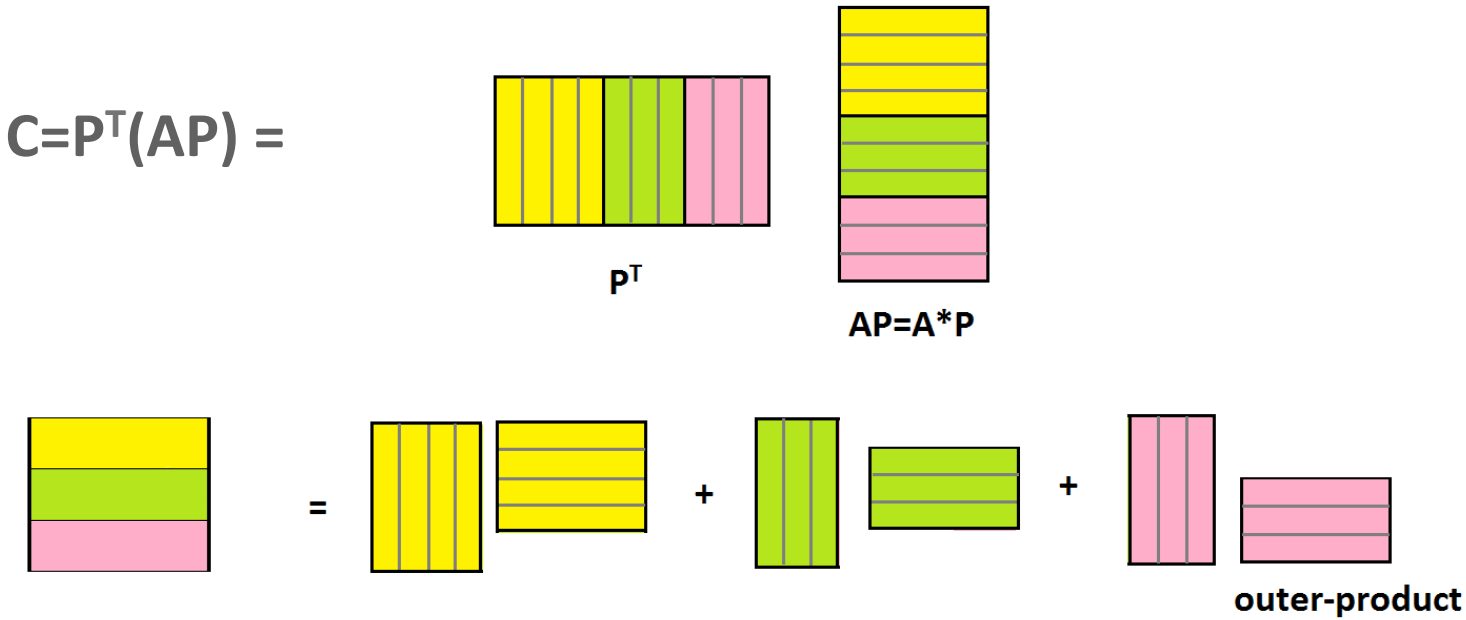


Weak Scalability of P^TAP
on Hopper@NERSC

# cores	27	216	1728	13,824
Solve time	13.2	17.2	19.5	
Setup	13.6	15.2	18.4	
P^TAP (5)	21.8	22.2	23.0	27.6
AB (5)				3.4
Iterations	39	50	55	
MFlops/core (solve)	248	245	241	

- AB only takes 12% of P^TAP execution time on 13,824 cores!

Sparse Matrix-Matrix Operations (parallel)



- Assembly C requires extensive inter-processor communications

Approach (suggested):

1. $R=P^T$ or user creates R
2. C=R*A*P

Sparse Matrix-Matrix Operations (parallel)

Trade-off between memory scalability and speed:

Example: C=A*P: C[i,:] = A[i,:] *Pseq

- '-matmatmult_scalable'

```
/* perform sparse axpy: apa_sparse: tmp array of local size Aloc_rmax*Ploc_rmax */
for (k=0; nextp<pnz; k++) {
      if (apJ[k] == pj[nextp]) { /* column of AP == column of P */
        apa_sparse[k] += valtmp*pa[nextp++];
      }
 }
```

- faster, but non-scalable:

```
/* perform dense axpy:  apa_dense: tmp array of global size PN */
for (k=0; k<pnz; k++){
      apa_dense[pj[k]] += valtmp*pa[k];
 }
```