# Parallel Bipartite Matching for Sparse Matrix Computations

Jason Riedy ejr@cs.berkeley.edu
Dr. James Demmel
UC Berkeley

## 1. Goal: Scalable Direct Linear Solver

- Numerical factorization scalable from static pivoting
  - Olschowka and Neumaier: Place large elements on diagonal
  - Modify any tiny pivots during factorization
  - Demonstrated by distributed SuperLU (Li, Demmel)
- Static pivoting is weighted bipartite matching
  - Augmenting path algorithms not parallel
- Can this be made 'scalable'?
  - This is a fast pre-processing step. Heavy lifting not allowed.
  - Scalable is too high a target.
  - *Distributed*, however, is fine.

## 2. Matching by Auction Algorithm

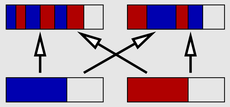
- Developed by Bertsekas, *et al.*
- Reduce to optimization problem:
  - Maximize $\mathrm{Tr}\, A^T X$ over permutation matrices, $A \in \Re^{N \times N}$.
  - Dual: Maximize prices $p$ and profits $\pi$ such that
    $$p1^T + 1^T\pi \leq C - \varepsilon$$
  - Slackness criteria:
    $$x_{ij}(c_{ij} - u_i - v_j) \leq \varepsilon$$
- Places a bid, moving the prices and possibly ejecting a losing column from the matching
- Finds solution within $N\varepsilon$ of optimum.
  - Can start with large $\varepsilon$ and scale down.

## 3. Parallel Auctions

- Core loop is simple and completely local
  - Examines non-zeros adjacent to a column for the best deal
- Each processor runs a local auction to completion.
- Processors merge results, global losers re-matched
- Send only *changes*, not full price vectors
- Requires a special reduction-like operation

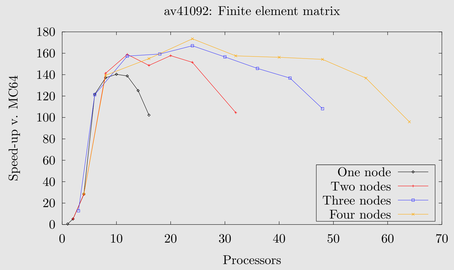
## 4. Sequence-Merging Reduction

- Simple max-reduction over price array: large slow-down
- Instead, merge the few price changes / bids
- Not supported directly by MPI collective communication
  - But same communication structure...
- Can work in butterfly pattern to reduce latency
  - Bids appear in different orders, but have same winners
  - Resolve ties by bidder (processor) number

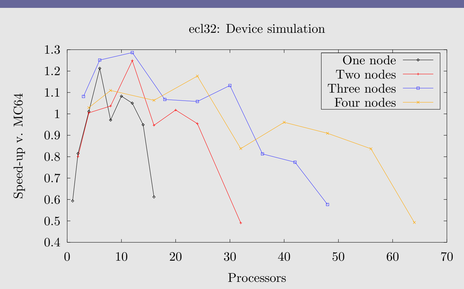


## 5. Irregular Problem, Irregular Speed-up

- Results from IBM SP/2 at NERSC (seaborg)
- Amazing (ridiculous?) speed-ups
  - 8 MB caches quickly hold whole matrix
  - Searching many paths through optimization space
  - Roughly $3-7\times$ less speed-up on 128KB cache Pentium 3s, but same curve shapes
- Sequential auction and MC64 (Duff, Koster) speeds are comparable.
- Using MPI; performance drops when 16-way nodes start to fill up.



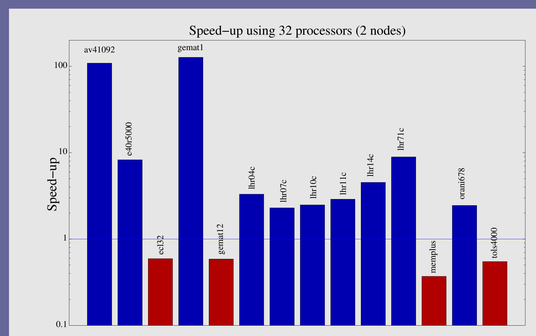

This speed-up goes beyond cache effects. The auction resolves very similar bids more quickly in parallel.

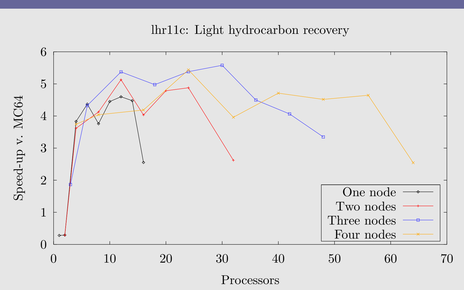



The `ecl32` matrix has a trivial maximum matching. The result is a measure of communication overhead.

## 6. So, Scalable?

- Nope, but well distributed
  - Does not require whole matrix on any processor
- Speed does not increase linearly with processors...
  - Runs out of problem: Small non-zeros per processor
  - Trivial matchings eat full communication overhead
- Compared to factorization, time less important

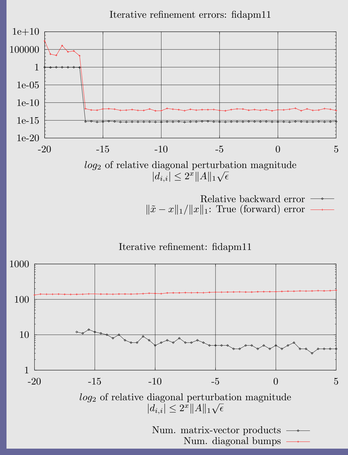


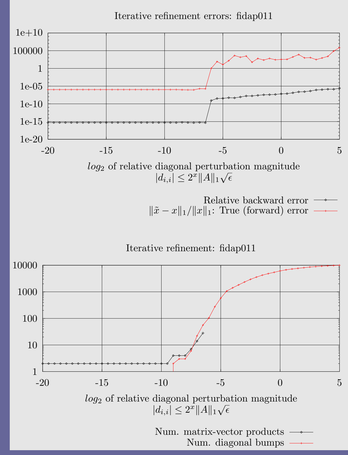

A less ridiculous speed-up from a 'typical' matrix.

## 7. Does Static Pivoting Work?

- Actually factoring $A + D$, $D$ low-rank and diagonal
- SuperLU: $|d_{i,i}| \leq \|A\|_1\sqrt{\epsilon}$
  - Here, $\epsilon$ is the machine precision parameter.
  - Plotted varied by $2^{-x}$...
- Using same value $d$ for threshold and $|D| \leq d$
- Without iterative refinement, results are mediocre.
- Convergence depends on threshold
  - Spectrum of $(A + D)^{-1}D$
  - Don't yet know if one $|d_{i,i}|$ bound works for all matrices
- Other iterative methods (*e.g.* GMRES(50)) often work where iterative refinement fails




- Small perturbation fails
- $A + D$ still numerically nearly singular




- Large perturbation fails
- $D$ too large relative to $(A + D)^{-1}$?

## 8. Future Work

- Experiment with shared memory
  - Local load balancing becomes free
  - Doesn't fit OpenMP well, needs sub-team barriers
- Making refinement converge
  - Is there a bound that works for all matrices?
  - Can we chose an appropriate bound during factorization?
- Determining convergence of other 'refinement' methods
  - GMRES(50) seems to always work.
  - Always expensive, sometimes extremely so.
  - How does performance vary with $D$ bound?
  - Can $D$ be adapted here?
- Try other pivoting strategies that don't change non-zero structure
  - Swap columns within a supernode?
  - Swap rows or cols within a front's non-update block?

TOPS Terascale Optimal PDE Simulations
ACTS Collection