Scaling SDPB Up and Down

Walter Landry
wlandry@caltech.edu

https://groups.google.com/forum/#!forum/bootstrap-collaboration-software

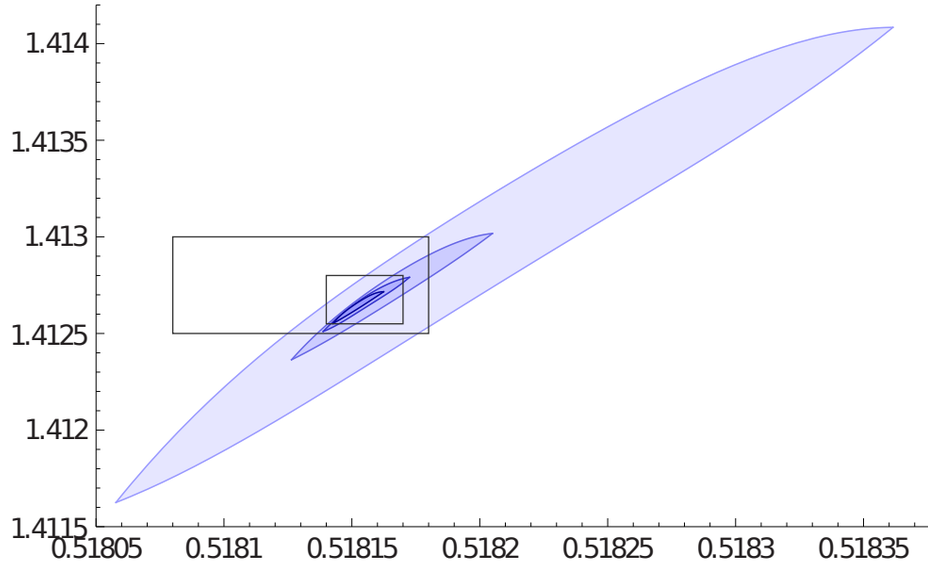
Caltech

The Bootstrap and Semidefinite Programs

- The conformal bootstrap can be formulated in terms of a semidefinite program.
- Semidefinite programs are generic math problems that occurs in many branchs of science and engineering.
- Existing, off-the shelf solver implementations exist in a variety of environments
  - Matlab, Mathematica, C, Python, ...

Why SDPB?

- Bootstrap calculations can require extreme numerical precision and computational resources.
  - Ising computations ran for weeks.



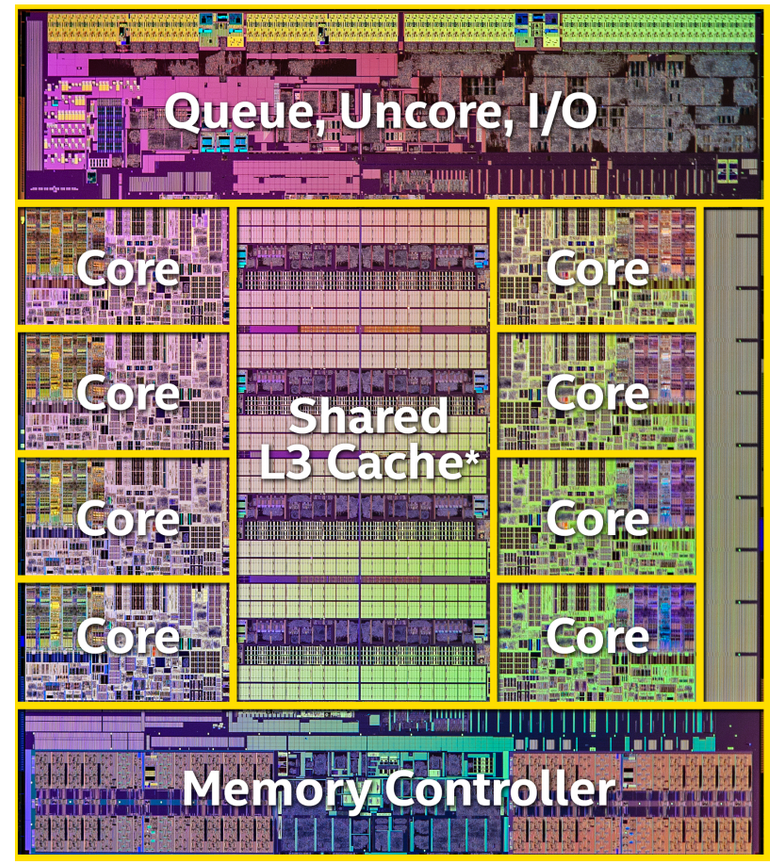
- SDPB is a solver optimized for bootstrapping.
  - Open-source
  - Arbitrary precision
  - Heavily parallelized

Parallelizing Linear Algebra

- Most of what takes time in SDPB is linear algebra operations on independent blocks of matrices.
- The results of these independent operations are combined into a single, comparatively small, global matrix Q.
- These independent operations can be run on different cores.

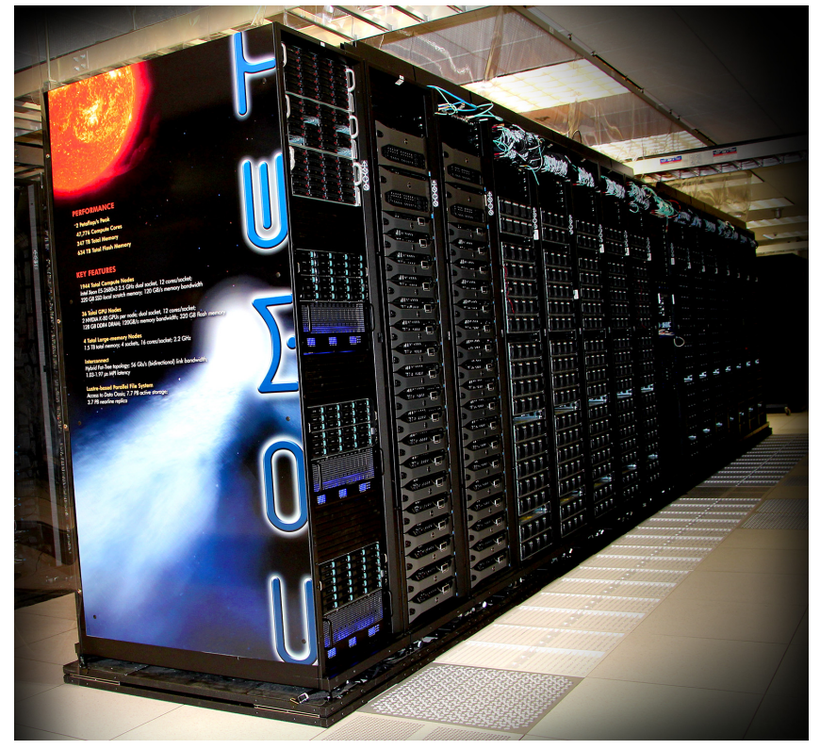
Parallelization with OpenMP

- SDPB was initially parallelized with OpenMP
  - OpenMP is very easy to use, but it relies upon a global view of memory.
  - Works on single nodes up to ~20 cores.
- Global view of memory quickly stops working beyond a single node.



Parallelism with MPI

- SDPB has been enhanced to use MPI (Message Passing Interface).
- MPI works by passing messages between cores.
- This works far better than OpenMP on supercomputers.
- It required extensive restructuring of the code.



Timing Runs

- Part of the restructuring is that we now have to explicitly assign these block computations to specific cores.

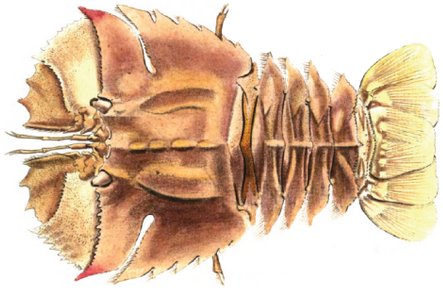
- Doing this well requires measuring how long it takes to run calculations for each block.
- Trying to derive the timings from first principles results in terrible performance.

Recent Work

- Automatic Timing
- Fake Primal Fix
- Faster Input
- Hot-starting and text checkpoints
- Installations
- Memory Use
- Scalar Blocks
- Spectrum Extraction
- Proposals

Automatic Timing

- SDPB now automatically performs a timing run.
- It is pretty transparent to the user.
  - No one asks me about timing runs anymore.



Fake Primal Solution

- There is a bug in the original, SDPB-OpenMP implementation of the primal error
- Paper says: $primalError = \max\{|p_i|, |P_{ij}|\}$
- Implementation was: $primalError = \max\{|P_{ij}|\}$
- **Usually** it makes no difference

- SDPB-MPI now reports both P and p, but it uses the full primalError for deciding when to stop.

Faster Input

- XML : pvm2sdp
  - Now fully parallelized
- SDP in Mathematica: sdp2input
  - Directly generates SDPB input files.
  - 16 times faster than SDPB.m on 28 cores
  - Enables some people to work without Mathematica (not all clusters have it).

Hot-starting is Fully Supported

- Allows you to start a new calculation with an older solution
- Can reduce the number of iterations by a factor of 10.

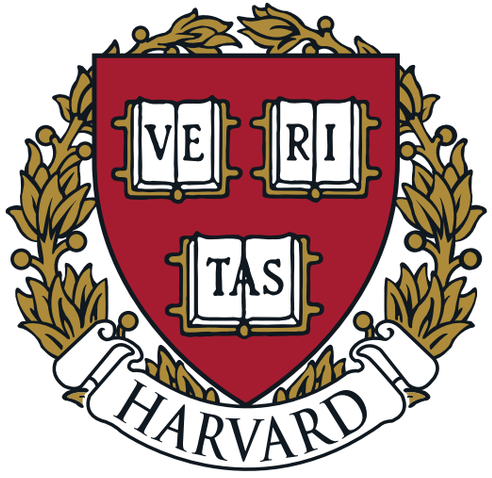
Text Checkpoints

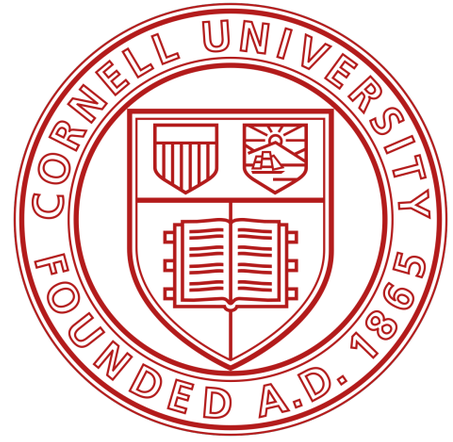
- Allows you to add to or modify an existing solution for a new problem.
- Portable across machines
- Not strictly bitwise identical.
  - The last bit can be different.
  - This comes from a limitation in the underlying GMP library.
  - It is unlikely to matter.

Easier Installation

- Better autodection of libraries
- No unnecessary dependencies.

Installed Everywhere



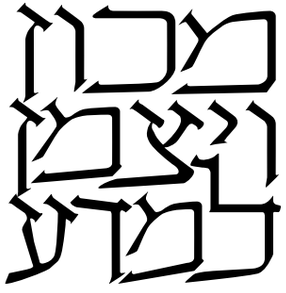


Caltech

EPFL

XSEDE

SDSC
SAN DIEGO SUPERCOMPUTER CENTER



Easiest Option

- For smaller runs on your laptop or desktop

  Docker (Windows, Mac, Linux) or

  
  Singularity (Linux: recommended)
- Download and Run
- Pretty efficient and uses all cores.
  - IAS admins used Singularity for their install on the Helios cluster.
- Instructions in the repository

  https://github.com/davidsd/sdpb/blob/master/docs/Docker.md
  https://github.com/davidsd/sdpb/blob/master/docs/Singularity.md

Much Better Memory Use

- Memory use is dominated by many cores having their own copy of their contribution to the matrix Q.
- Q is symmetric, so we now explicitly deallocate half of it.
  - The underlying parallel linear algebra library, Elemental, is not accustomed to this, so we have to be a bit careful.

procGranularity

- Added the option procGranularity
- Spreads the local contribution to Q across more cores
- A bit slower, so only use if desperate

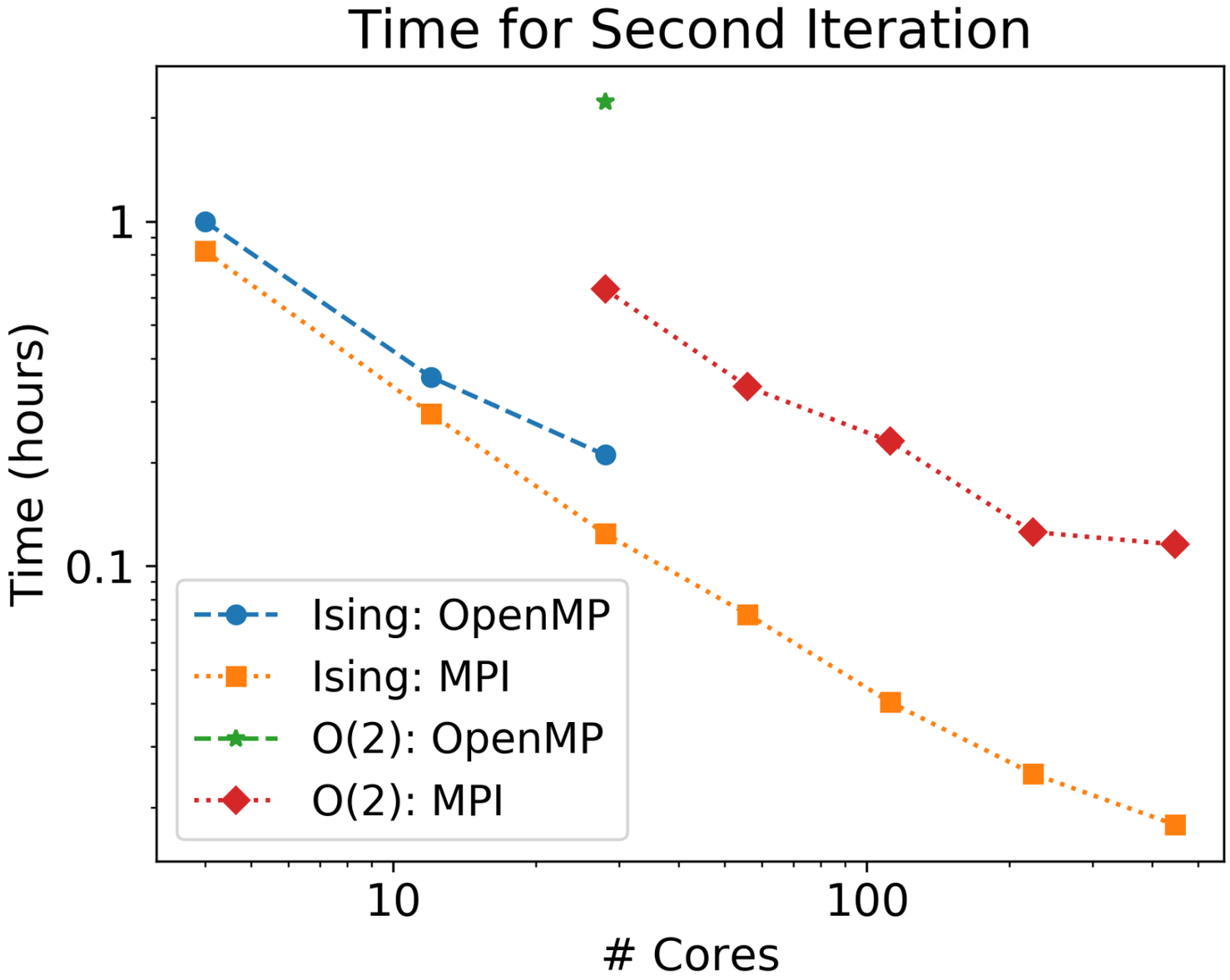
Synchronizing Q

- The local contributions to Q are summed and then distributed to a global Q with the low level routine MPI_Reduce_scatter.
- MPI_Reduce_scatter requires an additional copy of Q on each core.
  - Reimplemented to remove these copies
- With procGranularity, the memory overhead compared to SDPB-OpenMP should now be minimal.

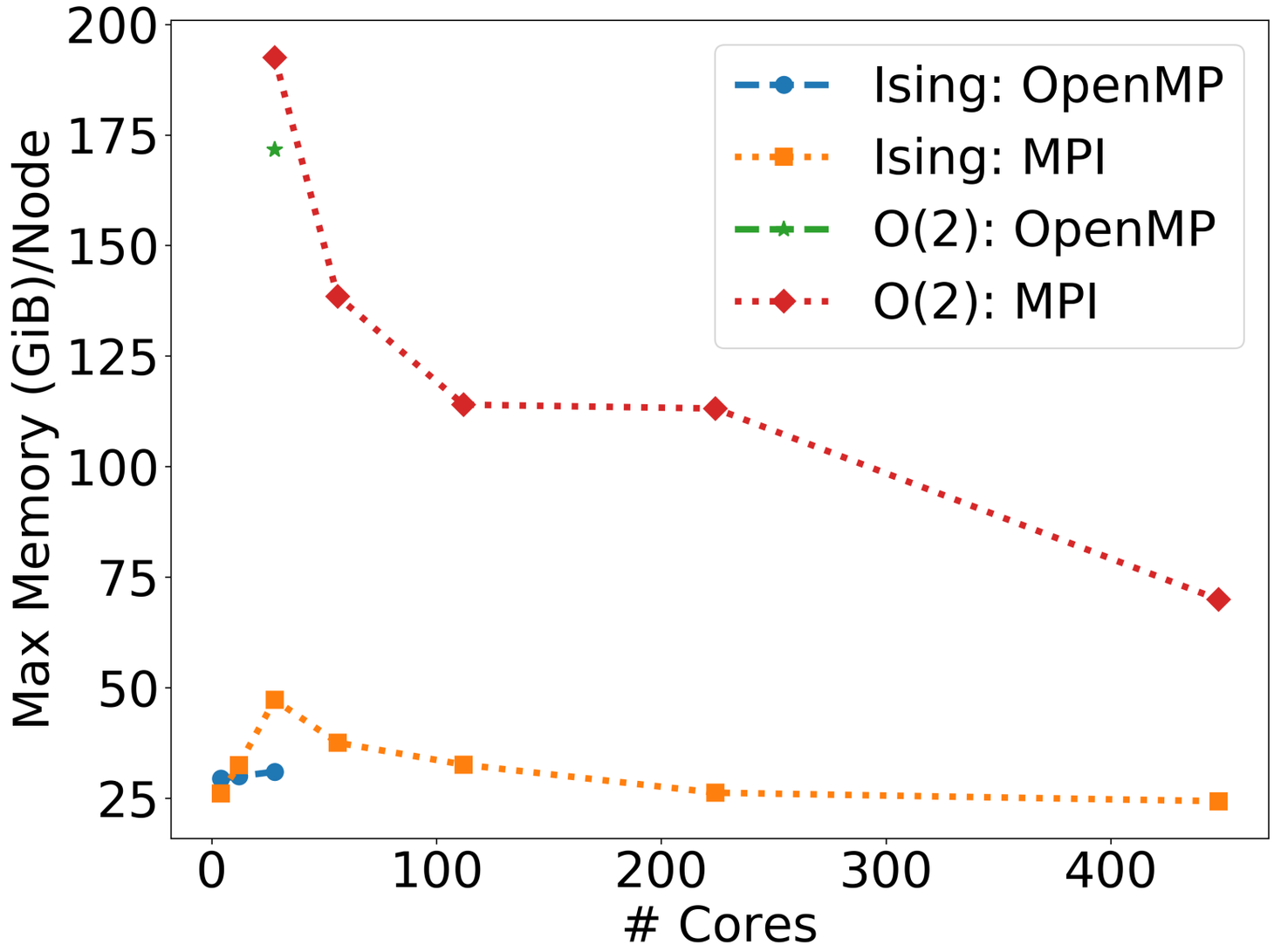
Q Caveats

- It is not as fast for large core counts.
  - Factor of 2-3 for O(2), n_max=18 with 448 cores at Yale
- However, you would only use large core counts for large problems.
- Previously, you would have a hard time fitting your large problem on the machine at all.
  - O(2), n_max=22 did not fit on Comet

Scaling on Large Machines



Memory Use


Max Memory (GiB)/Node
200
175
150
125
100
75
50
25
0
100
200
300
400
Cores
Ising: OpenMP
Ising: MPI
O(2): OpenMP
O(2): MPI

O(2) Remarks

- The O(2) project has been an excellent driver of progress for SDPB.
- It generated large, concrete benchmarks that people definitely wanted to solve.
- It highlighted bottlenecks when performing a complete bootstrap calculation, motivating improvements to block generation (scalar_blocks) and conversion from Mathematica SDP's to input (sdp2input)

Scalar Blocks

- Replaces Mathematica block generation
- Written in C++
- 111 times faster on 28 cores
- Available in the 

and  images.

https://gitlab.com/bootstrapcollaboration/scalar_blocks

Spectrum Extraction

- Python script to extract the spectrum from the SDPB output
- Updated for new output format
- Clarified dependencies and made to work with python 2 or 3
- Also available in the 

and  images.

https://gitlab.com/bootstrapcollaboration/spectrum-extraction

XSEDE Proposal

- XSEDE is an NSF funded clearinghouse for supercomputer time at different centers.
- We wrote a proposal for the O(2) project for 5 million hours on the SDSC Comet cluster.
- Awarded 3 million hours
- Received 1.2 million hours
- Used up 200,000 hours in a few days

Cannon Cluster Proposal

- Harvard is standing up Cannon, a new cluster with 30,000 cores.
- They are looking for users who can thoroughly exercise the machine.
  - Science would be nice, but is not the driver
- Request for Proposals: Up to 3 days of compute time on the whole cluster.
- We submitted a proposal for ~1 million hours for more O(2) work.

Ongoing Work

- Scaling
- Precision

Better Scaling

- The work so far has pushed the scalability of SDPB from ~20 cores to ~300.
- We have run jobs up to 768 cores.
- The rule of thumb is that each improvement by a factor of 10 takes significant effort.
- The next step will require careful benchmarking on large machines.

Why Such High Precision?

- I will be looking at a small stress tensor example. It seems non-trivial enough to be useful.
- You might expect to need only to resolve
  - The error threshold: 10^{-40}
  - The duality gap between the primal and dual solutions: 10^{-80}
- In practice, we need much, much higher precision.

What Breaks?

- The first thing that breaks when reducing precision is when solving

$$\begin{pmatrix} S & -B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} r_x \\ r_y \end{pmatrix}$$

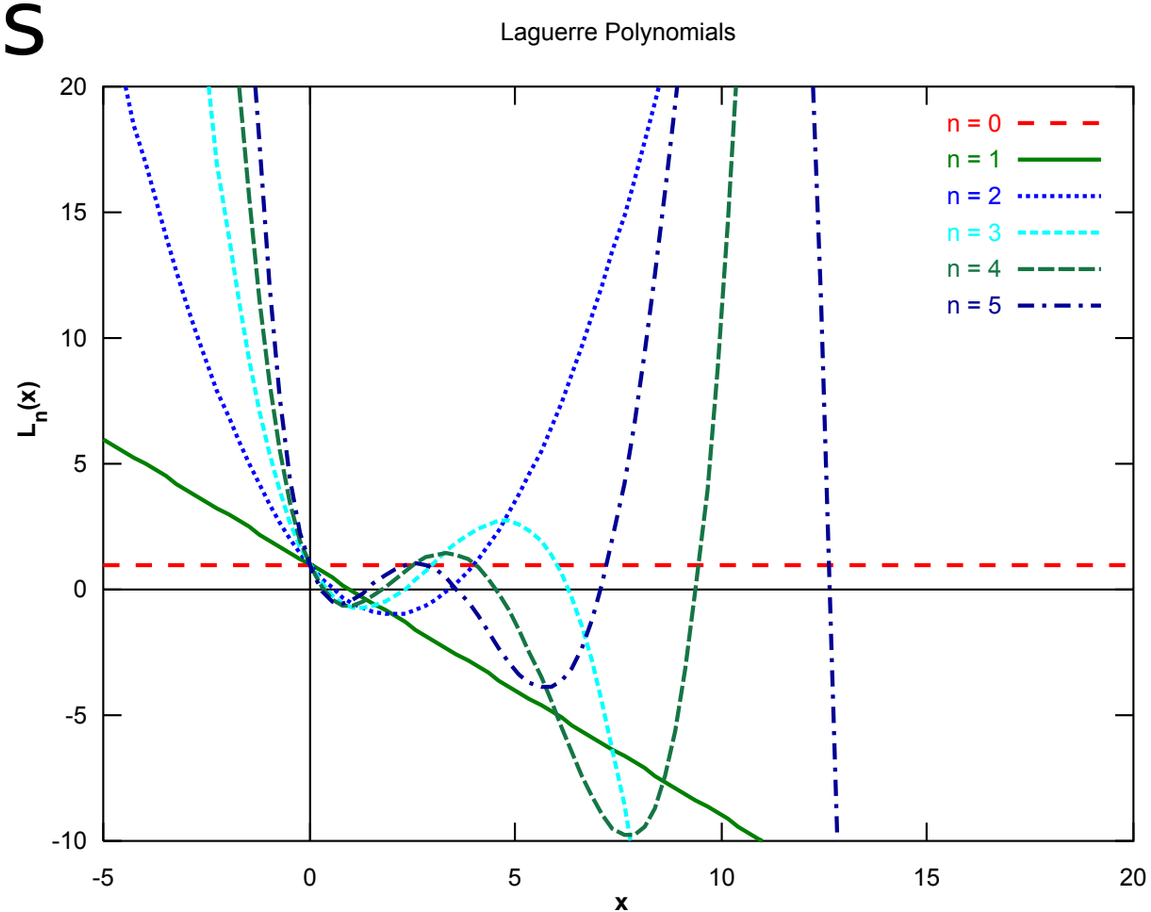
- S has a block structure made up of symmetric positive-definite matrices.
- We use a Schur complement method, which involves inverting S first.

S is Ill-Conditioned

- When precision is low, S is no longer numerically positive.
- This is because S has a very bad condition number: 10^{180}
- This happens immediately, well before we do any real calculations.

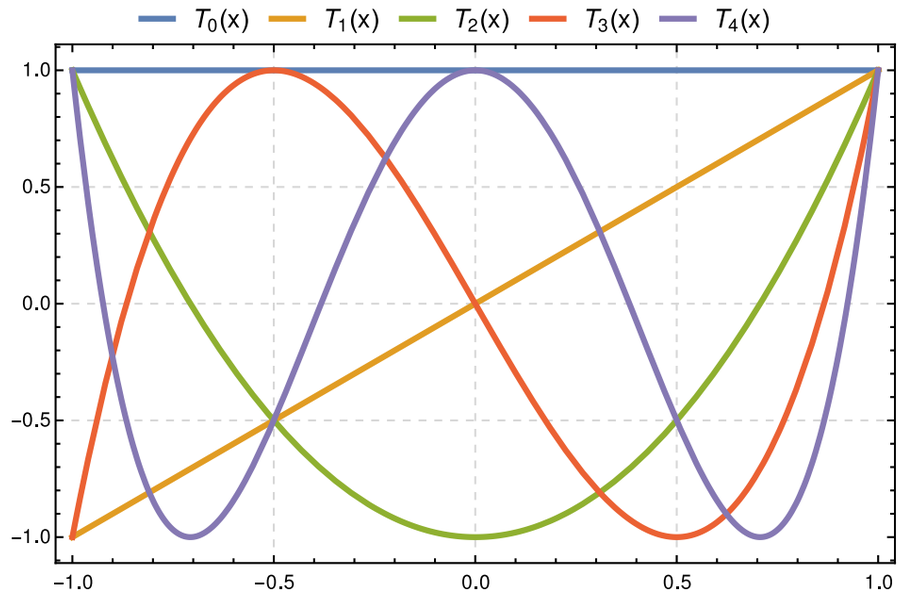
Bad Basis

- By default, we evaluate functions at the roots of Laguerre polynomials.
- Laguerre polynomials mimic exponentials, but the function we are approximating is well behaved over the domain.



Chebyshev Polynomials

- Chebyshev polynomials are ***very*** well behaved in their domain.
- We tried mapping the Chebyshev roots to the same interval.
- Evaluating functions at these new points yields a dramatic improvement in the condition number of s: 10^4

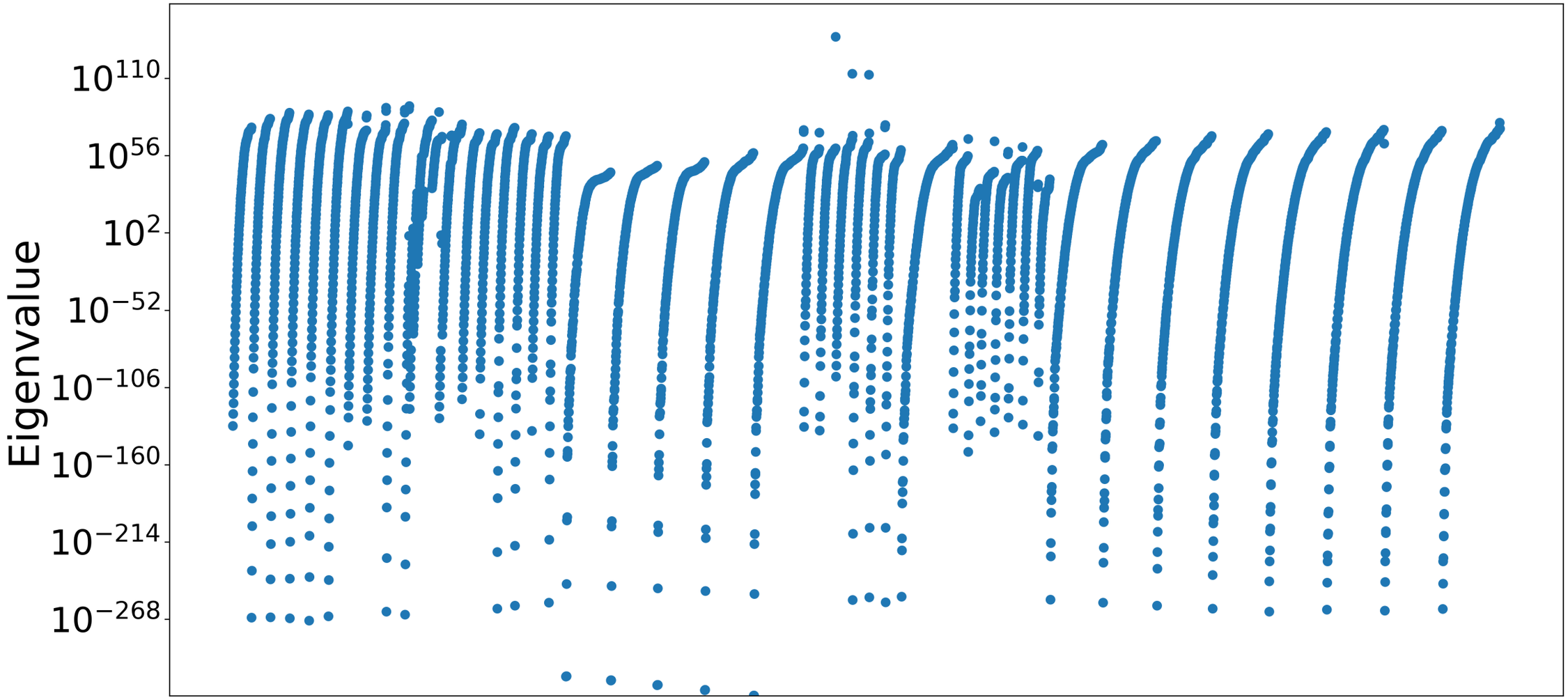


Not the Solution

- Unfortunately, as SDPB converges on a solution, S again becomes very, very ill-conditioned:

$$10^{400}$$

Eigenvalue Spectrum of S



Eigenvalue Spectrum

- The regular structure is from each of the blocks of S having their own range of eigenvalues.
- The eigenvalues smoothly vary from miniscule to gigantic. There is no natural break.

Eigenvector Decomposition

- If we decompose dx and B into eigenvectors of S, it turns out that:

$$dx_i \propto \lambda_i^{-1/2} \qquad B_i \propto \lambda_i^{1/3}$$

- This implies $r_y = B \cdot dx \propto \lambda_i^{-1/6}$
- But r_y lives in a different space and at this point in the calculation, after a dual jump, is essentially zero.

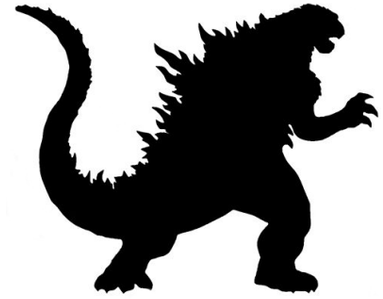
Small Differences - Big Problems

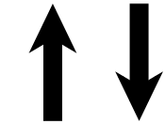
- The scaling eventually breaks down at small λ_i, but there are still many cancellations over a large range of λ_i.
- It does mean that we can not just ignore small eigenvalues.
- So there is still more to understand.

New Work

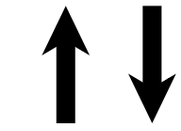
- Gateways
- Cloud
- Job Management

Gateway





THE APACHE SOFTWARE FOUNDATION



- A web interface to SDPB
  - Simple pointy-clicky
  - Scriptable (https POST)
- You upload input files.
- The Gateway submits these files to a supercomputer.
- You check progress from time to time.
- When finished, you collect the results

Gateway Users

- This removes the need to understand supercomputers in order to do large calculations.
- Even for those who do understand them, there is no need to get any complex control software running there.
  - For example, Mathematica is not available on XSEDE

Gateway Implementation

- XSEDE is very interested in giving away free time for gateways.
  - I have implemented a gateway with them in the past (seismology simulations)
  - I have also run high traffic web sites with scientific users.
- We would essentially become a mini-funding agency, giving out SDPB time.
- Requires a committee to vet applications

Scaling to the 

- Many different, large providers
- You can make use of enormous compute power.
  - Western Digital burned 8 million core-hours in 8 hours on hard drive simulations.
- Caltech seriously considered using the cloud instead of building their own supercomputer.

Cloud Details: $$$

- It costs serious money, but sometimes they give away time for free
- Compute is cheap-ish: 2-3 cents/core-hour
  - The Western Digital runs cost $137,307
- Storage is not cheap: ~2 cents/GB-month
- Transferring data into the cloud is free.
  - Getting data back out is not: ~9 cents/GB
- Good match for SDPB
  - long calculations and small outputs

Cloud Implementation

- Smaller, 1-node jobs may already work with Docker?
- Larger jobs require more thorough investigation and performance testing.
  - AWS ParallelCluster
    - No Infiniband. Maybe EFA is not terrible?
  - Azure Batch
    - Requires Intel compiler?
  - Not as big a market, so fewer choices

Managing Computations

- Assuming that all of these resources are available, we still need a way to manage all of the separate computations that combine into a single result.
- Existing approaches use a variation of Mathematica scripts or Haskell programs.
- We should figure out what is common to all of these approaches and automate that.