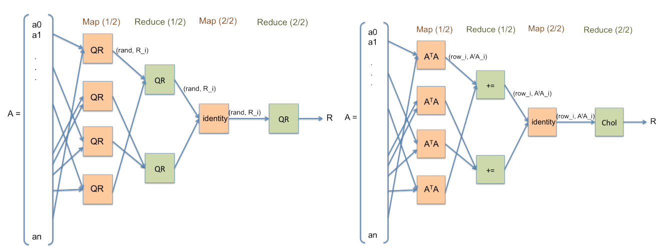
Berkeley Parlab

MapReduce TSQR

Austin R. Benson
arbenson@berkeley.edu

TSQR

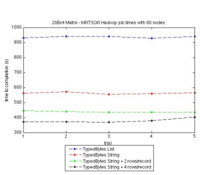
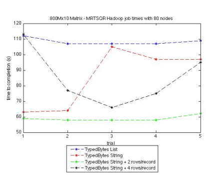
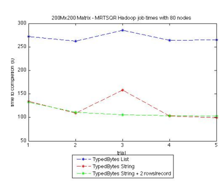
- $A = QR$, $Q^TQ = I$, R upper triangular
  - Many more rows than columns → "tall and skinny" (TS)
  - Lots of embarrassingly parallel work
- Two methods for computing R
  - "TSQR" algorithm by Demmel et al. [5] (slower, more stable)
  - Cholesky decomposition on A^TA (faster, less stable)
  - Both algorithms scale well
- Make these algorithms run fast in the cloud [1], [4], [6]
  - Implementations with Apache Hadoop MapReduce
  - How does numerical stability factor in?



MapReduce schemes for TSQR (left) and Cholesky QR (right)

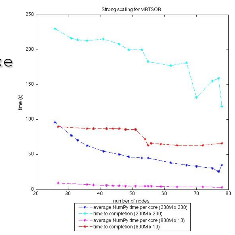
Data Serialization

- TypedBytes storage [2], [3]
  - Different data types in sequence file
  - String format yields great improvement over list format
- Packed rows
  - Store 2 or 4 rows per record
  - Performance can be significantly better, stay the same, or get worse

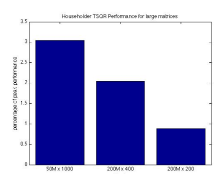
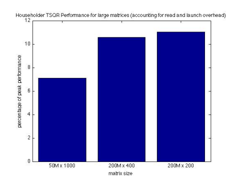
Strong Scaling

- Difficult to control processor allocation
  - In Hadoop, use `mapred.min.split.size` parameter
  - Embarrassingly parallel work greedily consumes any extra resources
- For TSQR, computation time small
  - Hadoop overhead and disk reads dominate time



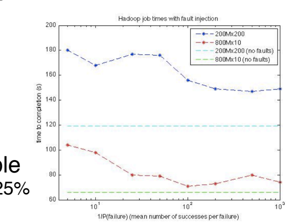
Peak Performance

- How close to peak performance?
  - 1-3% peak for large matrices
  - For a MapReduce architecture, this is about what we expect
- Refining the model
  - ~60 seconds for launch, cleanup overhead
  - ~60 MB/s disk reads (1 TB SATA disks)
  - → 7-11% peak

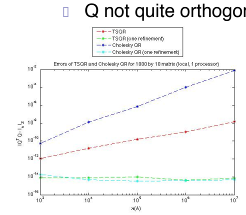
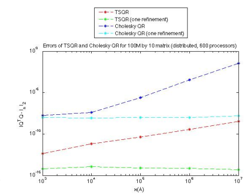
Fault Tolerance

- Key advantage to Hadoop and other MapReduce architectures
  - How does this affect performance?
  - How does P(fault) affect performance?
- Noticeable but manageable
  - Faults quickly introduce a 25% performance hit
  - P(fault) ~= 1/5 → only 50% performance hit



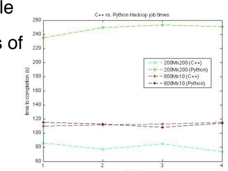
Numerical Experiments

- How stable are our algorithms?
  - Compute R with TSQR and Cholesky
  - $Q = AR^{-1}$, check $\|Q^TQ-I_n\|_2$
  - Q not quite orthogonal → $A = QR = Q'R'R$

Streaming: C++ vs. Python

- Use Hadoop streaming
  - Python provides easy prototyping for testing algorithms at a large scale (both algorithms implemented in 100 lines of code)
  - C++ can give us better performance
- Number of columns matters
  - 10 columns → about the same performance
  - 200 columns → C++ runs 4x faster than Python



Future work

- Implement customized data storage
- Expand to other areas of linear algebra (e.g., LU) and tensor algebra
- Explicit formulation of Q
- Experiment with other MapReduce frameworks (e.g., Spark, Twister)

References

[1] Austin Benson. MapReduce TSQR code. https://github.com/arbenson/mrtsqr
[2] Klaas Bosteels. Dumbo. https://github.com/klbostee/dumbo
[3] Klaas Bosteels. TypedBytes. https://github.com/klbostee/typedbytes
[4] Paul G. Constantine and David F. Gleich. Tall and skinny QR factorizations in MapReduce architectures. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 43-50, New York, NY, USA, 2011. ACM.
[5] James Demmel, Laura Grigori, Mark F. Hoemmen, and Julien Langou. *Communication- optimal parallel and sequential QR and LU factorizations*. UCB/EECS-2008-89. August 2008.
[6] David Gleich. MapReduce TSQR code. https://github.com/dgleich/mrtsqr