MapReduce TSQR

Berkeley Parlab

Austin Benson

arbenson@berkeley.edu

Author

Author

Author

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)

Make these algorithms run fast in the cloud [1], [4], [6]

- Benchmark performance using Apache Hadoop
- How does numerical stability factor in?

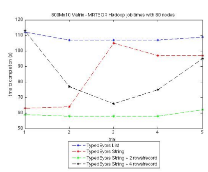
Data Serialization

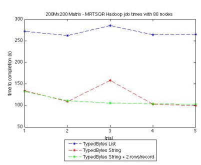
TypedBytes storage [2], [3]

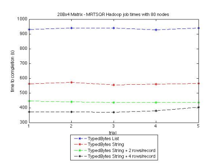
- Different data types in sequence file
- String format yields great improvement over list format

Packed rows

- Store 2 or 4 rows per record
- Performance can be significantly better, stay the same, or get worse







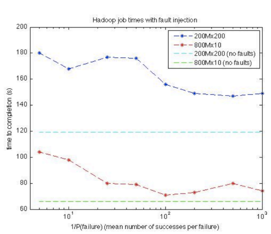
Fault Tolerance

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!



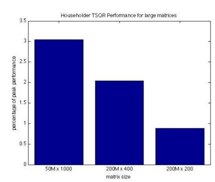
Peak Performance

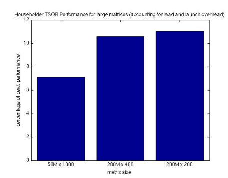
How close to peak performance?

- 1-3% peak for large matrices
- For a MapReduce architecture, this is about what we except

Refining the model

- ~60 seconds for launch, cleanup overhead
- ~60 MB/s disk reads (1 TB SATA disks)
- → ~7-11% peak





Numerical Experiments

Text Text

- Text
- Text

Text Text

- Text
- Text

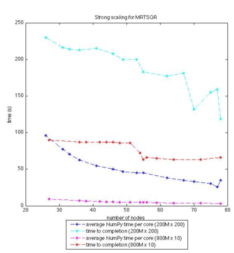
Strong Scaling

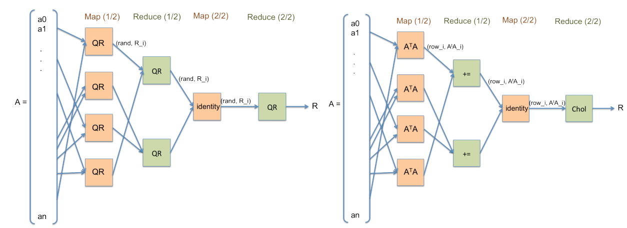
Difficult to control processor allocation

- In Hadoop, use `mapred.min.split.size` parameter
- Embarassingly parallel work greedily consumes any extra resources

For TSQR, computation time small

- Hadoop overhead and disk reads dominate time





MapReduce schemes for TSQR (L) and Cholesky (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

Text Text

- Text
- Text

Text Text

- Text
- Text

Future work

- Implement customized data storage
- Expand to other areas of linear algebra (e.g., LU)
- 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. MAPREDUCE 2011.*

[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