MapReduce TSQR

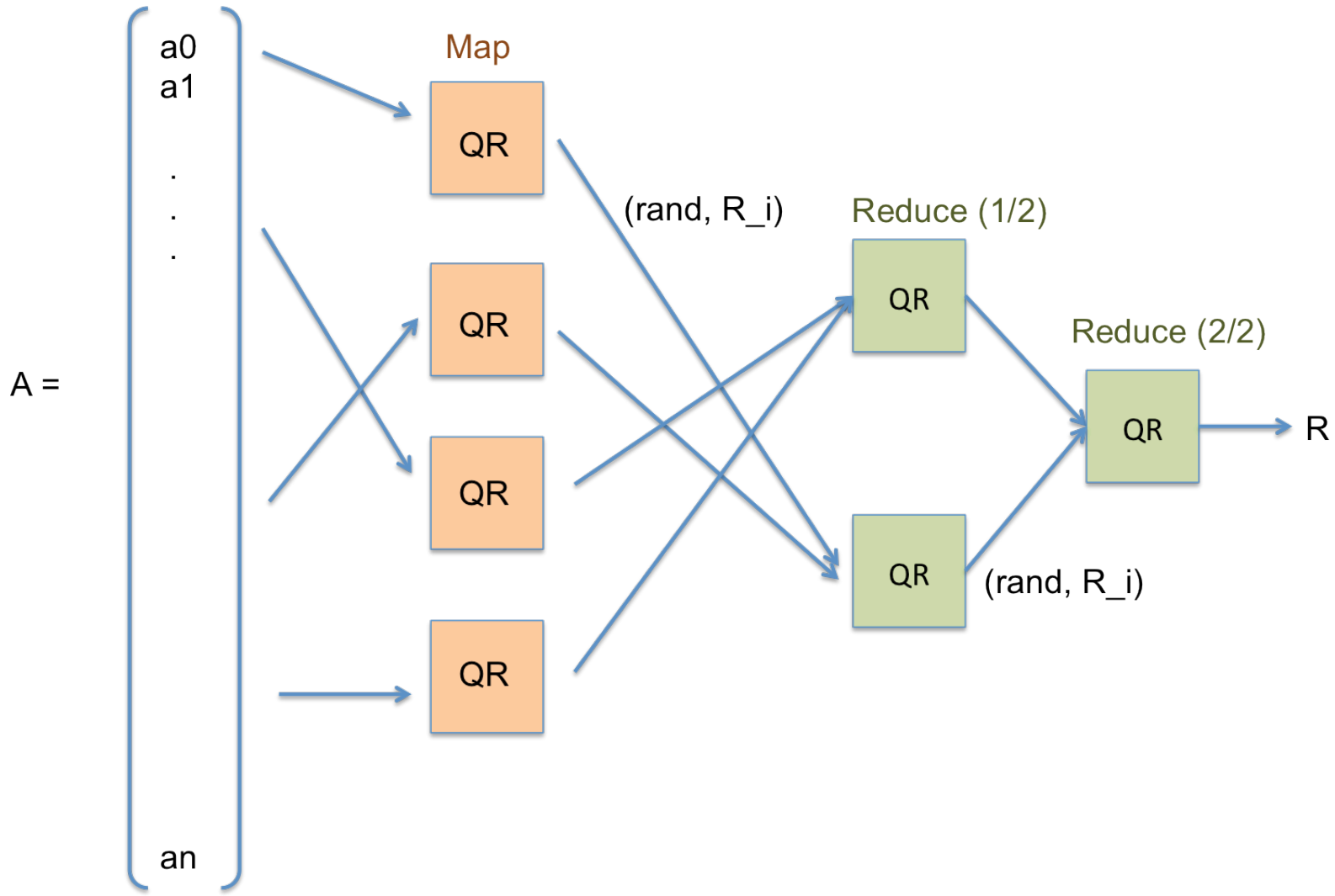
Austin Benson

Current benchmarking

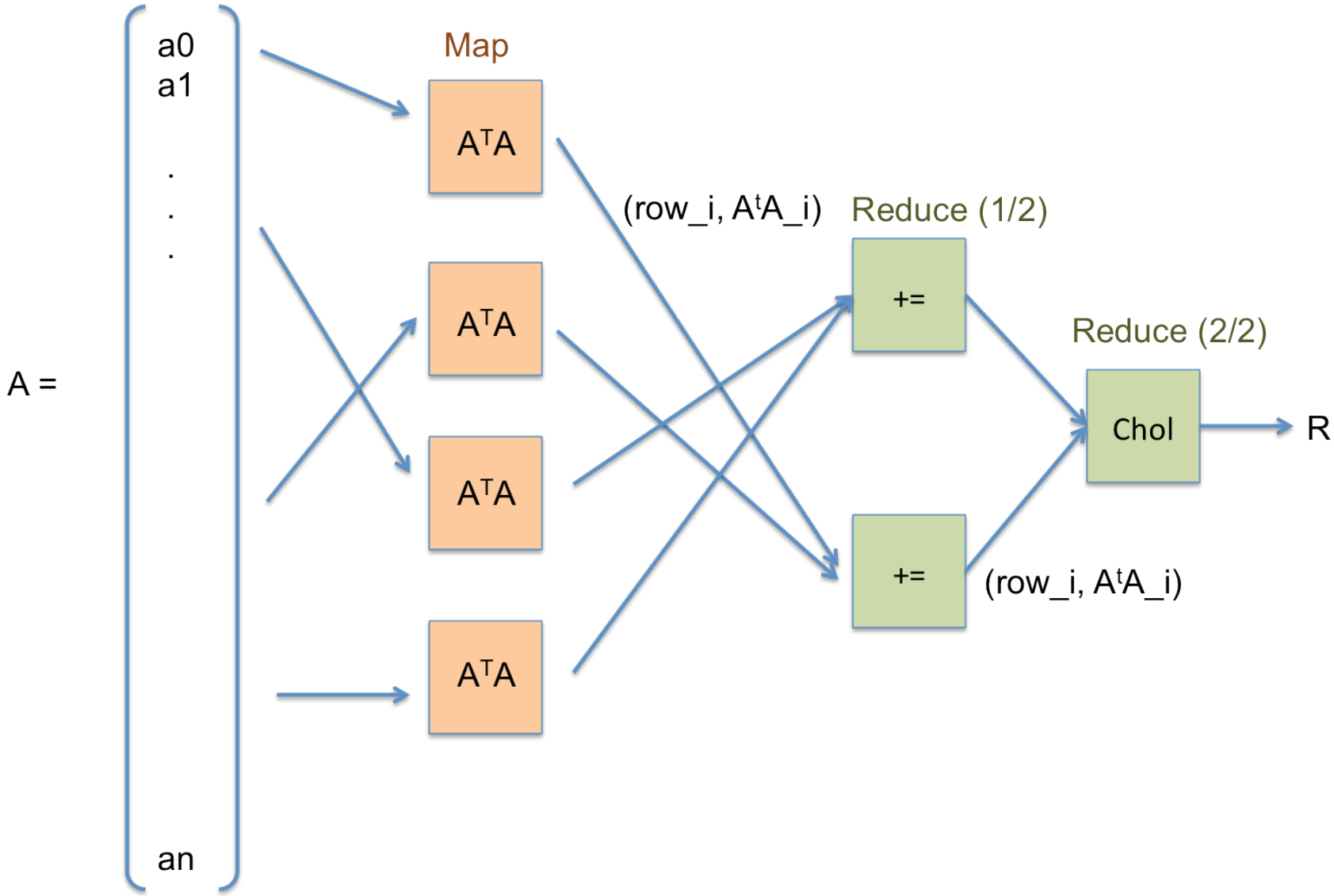
- Different data types for Hadoop Streaming input records: use a byte sequence instead of an array sequence via python's struct module
- Store multiple rows in one record
- Measuring variability across trials
- Simulating faults (via software) and relating fault rate to performance
- Python vs. C++ for Hadoop Streaming
- Cholesky QR vs. TSQR algorithm

See:
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
http://www.cs.purdue.edu/homes/dgleich/publications/Constantine%202011%20-%20TSQR.pdf

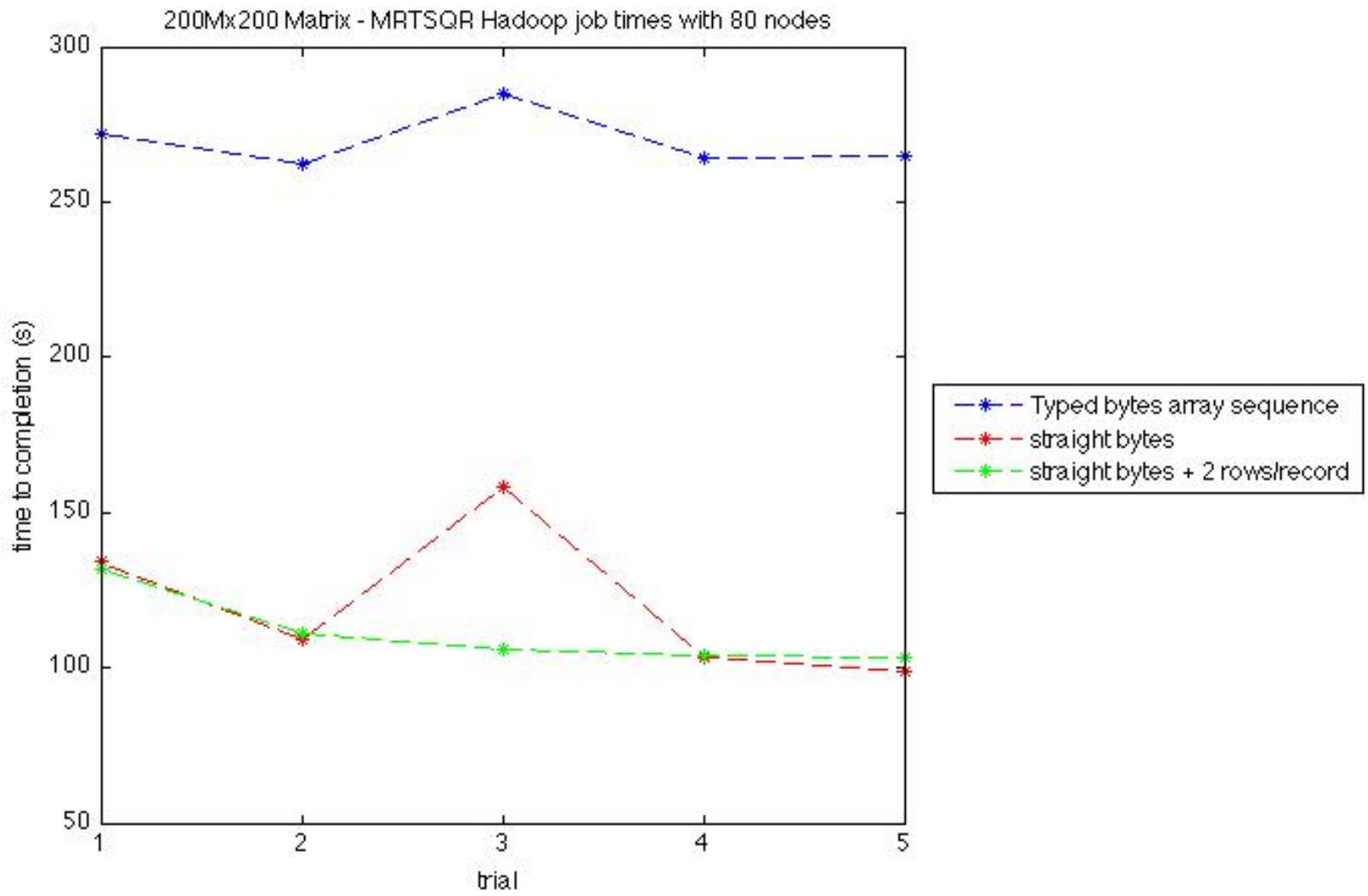
TSQR Scheme


A =
a0
a1
.
.
.
an
Map
QR
QR
QR
QR
(rand, R_i)
Reduce (1/2)
QR
QR
(rand, R_i)
Reduce (2/2)
QR
R

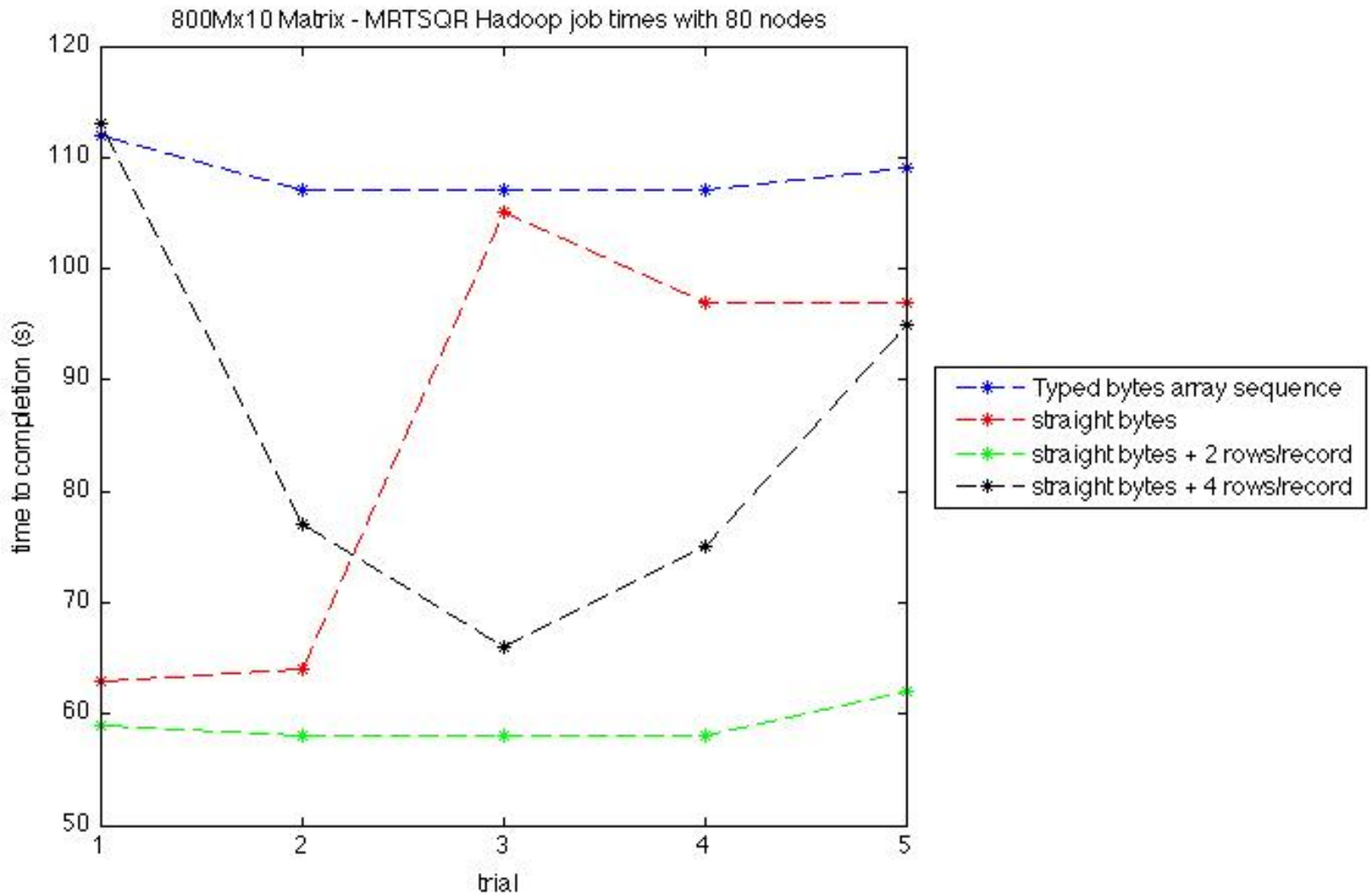
Cholesky QR Scheme



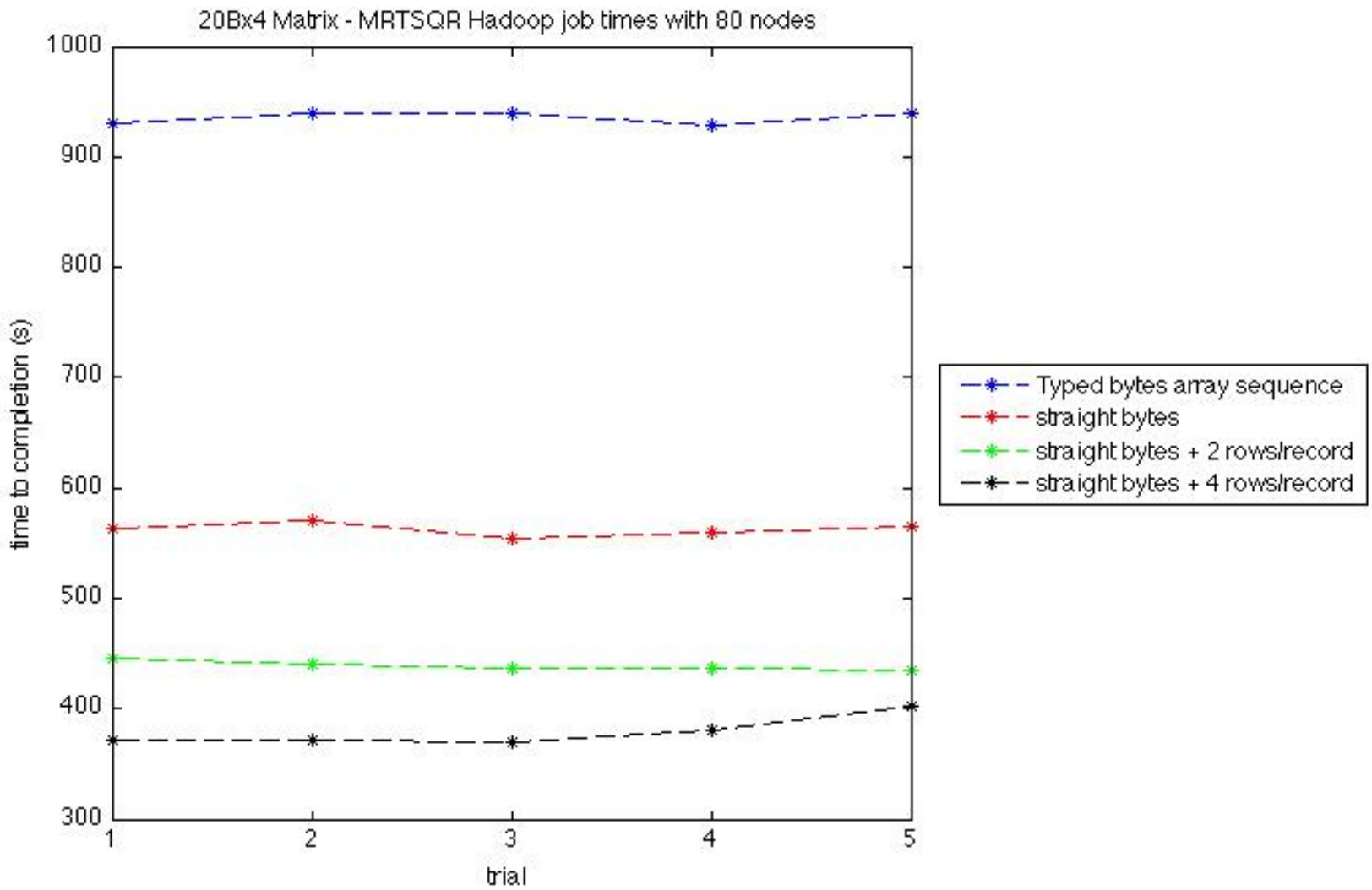
Performance results (200M x 200)



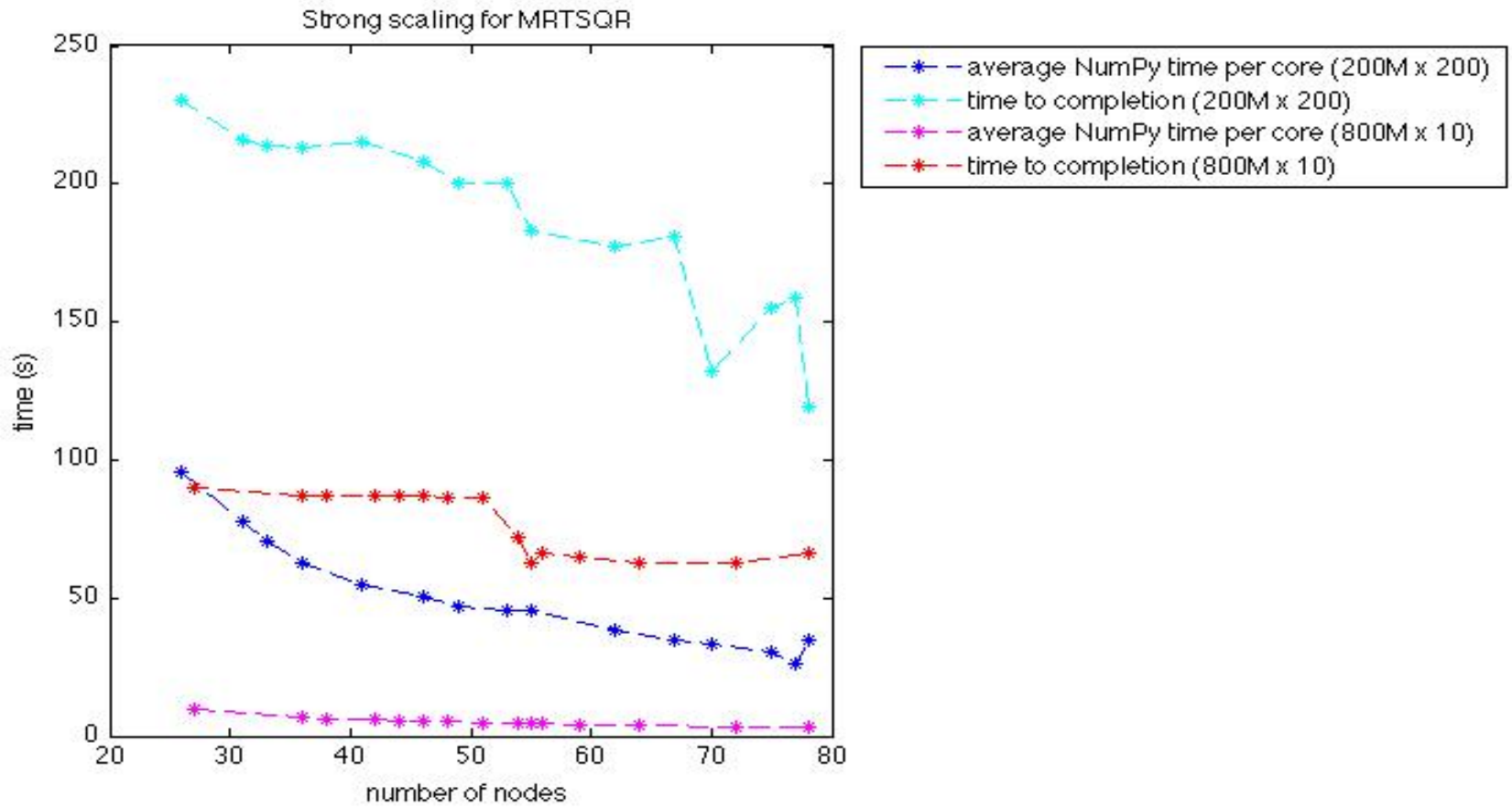
Performance results (800M x 10)



Performance results (20B x 4)

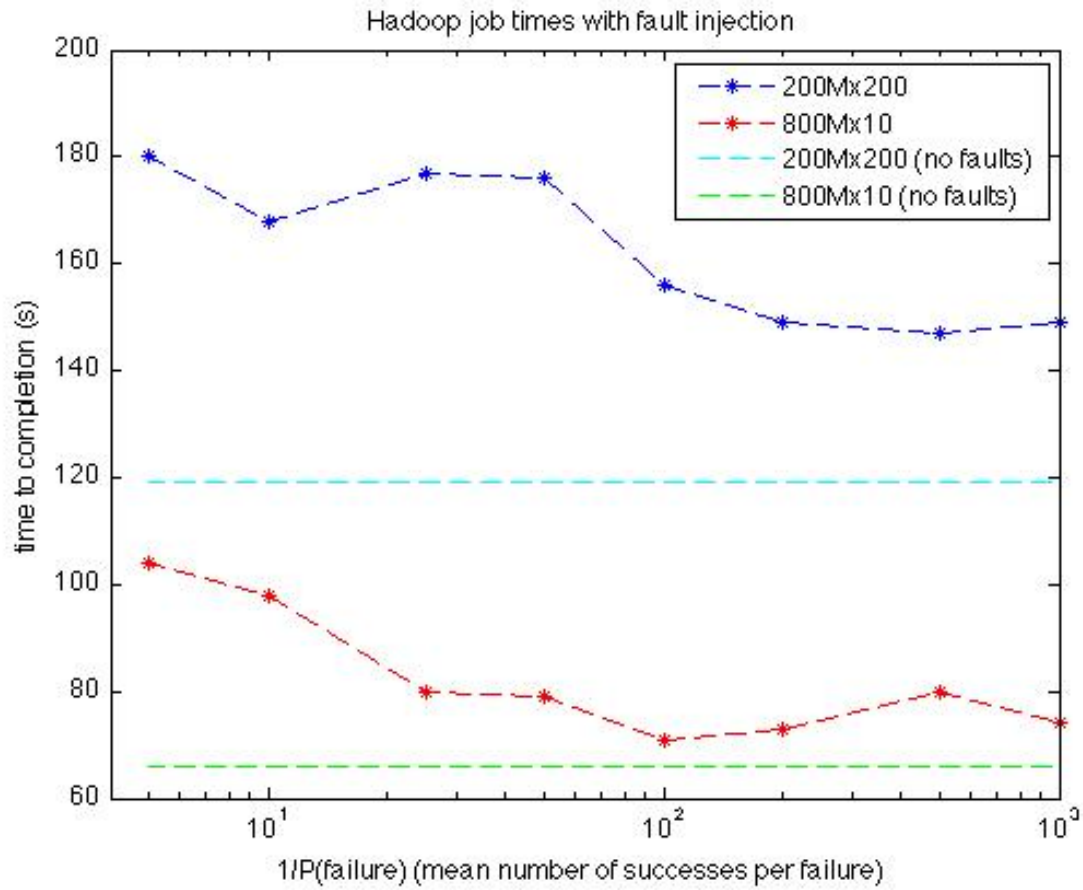


Performance results (strong scaling)



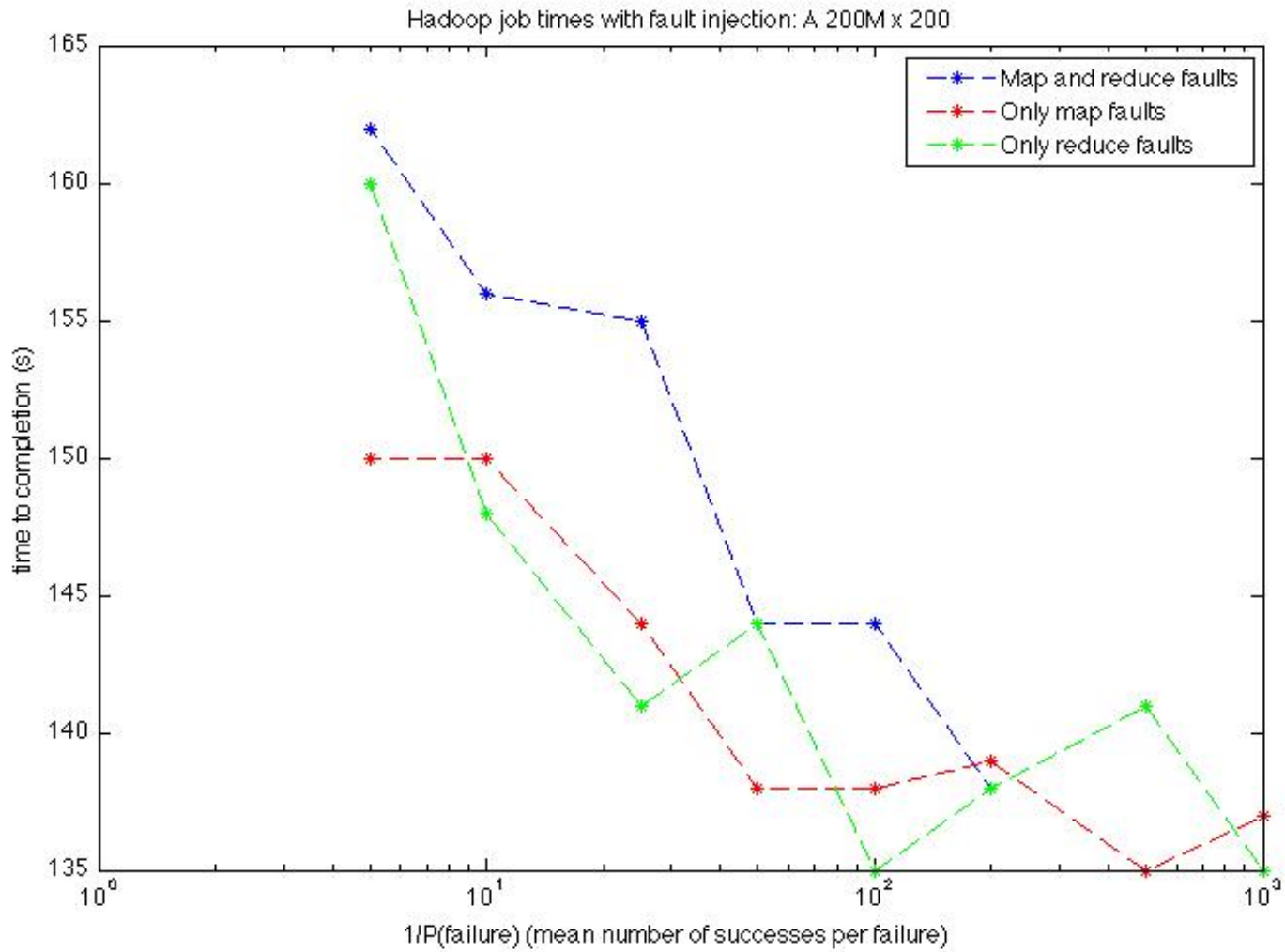
Hard to scale number of nodes in cloud environment--accomplished by tuning the sensitive mapred.min.split.size parameter

Performance results (simulated faults)

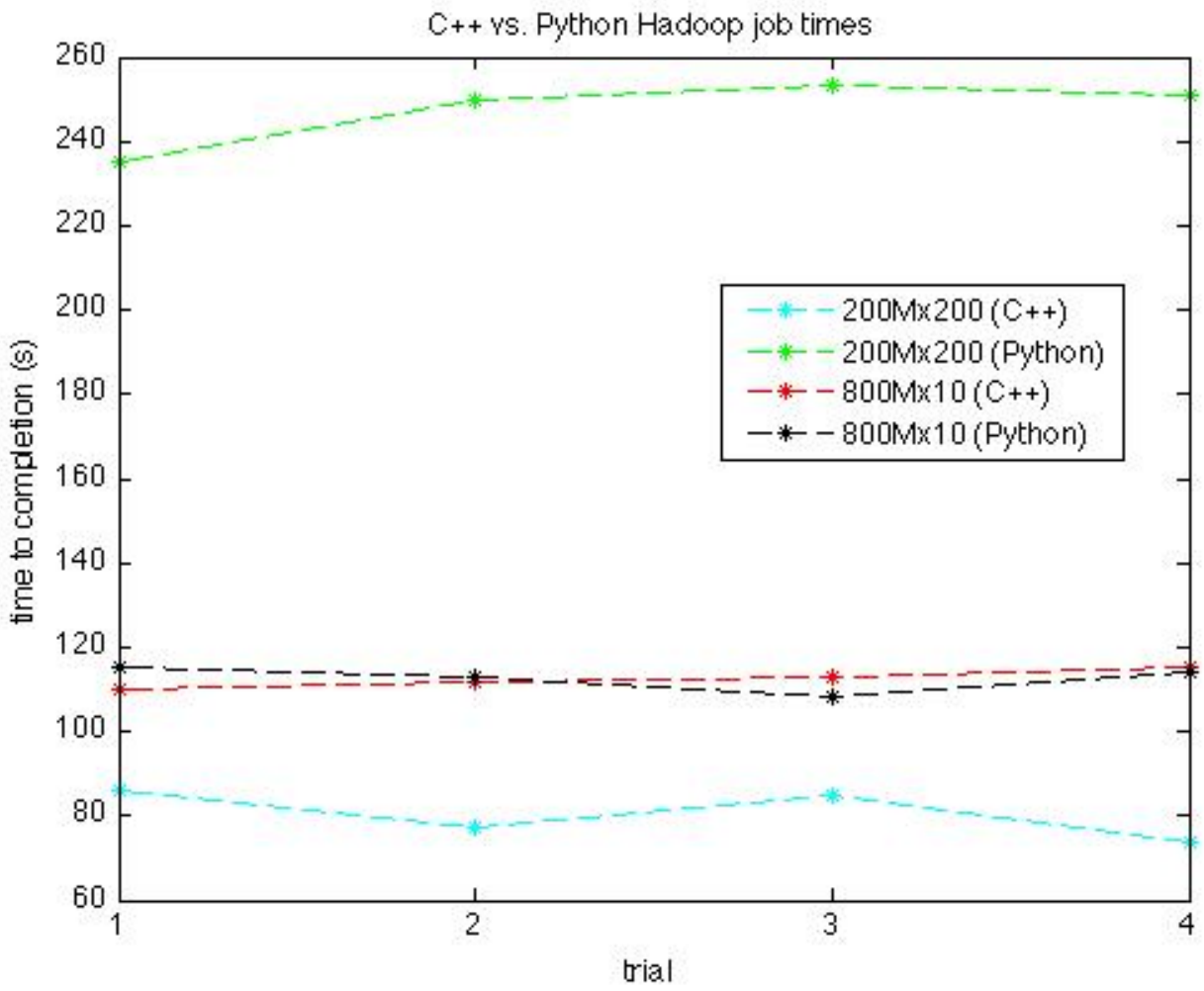


We can still run with P(fault) = 1/5 with only ~50% performance penalty. However, with P(fault) small, we still see a performance hit.

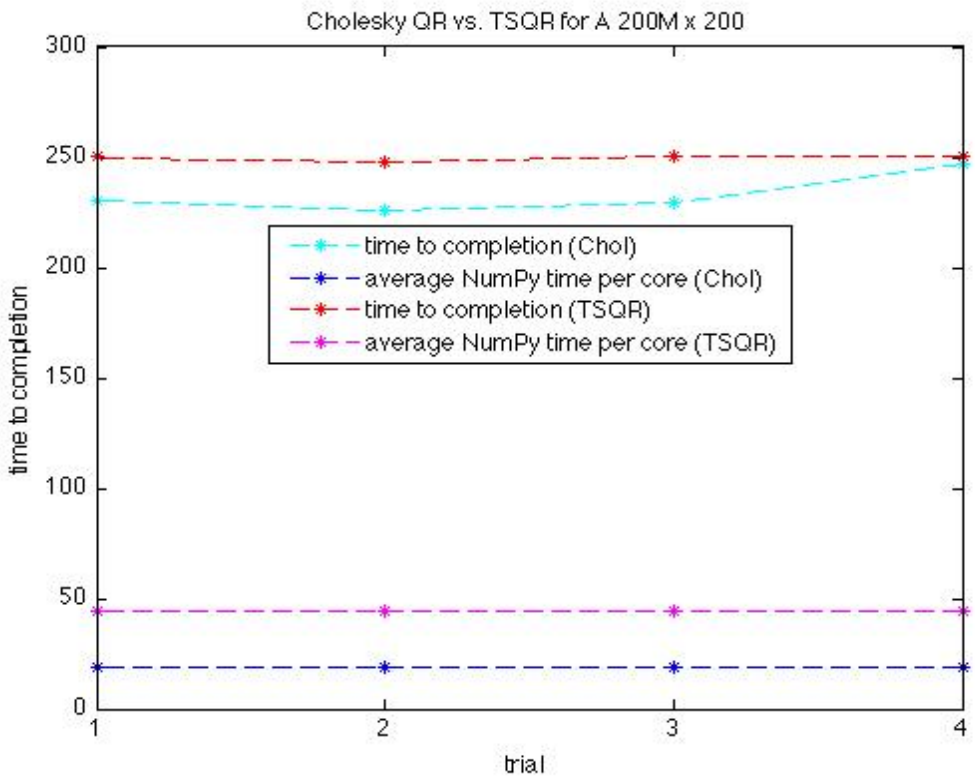
Performance results (simulated faults)

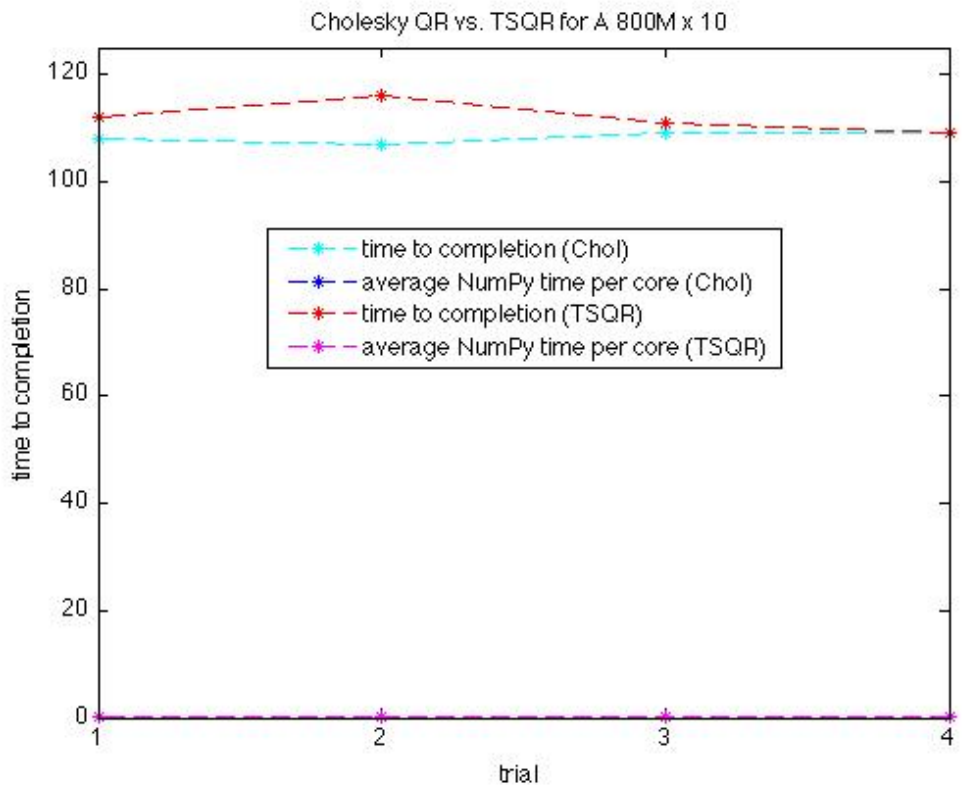


Performance results (C++ vs. Python)



Performance results (CholQR vs. TSQR)*





* This is just measuring map time, due to a bug in the Cholesky code. Reduce time is much smaller and constant.

Future work

- Custom input/output reader/writers
- More C++ implementation performance results
- More Cholesky QR data
- Numerical stability experiments