Transactions and Privatization in Delaunay Triangulation

Michael L. Scott, Mike Spear,
Luke Dalessandro, and Virendra Marathe

University of Rochester
www.cs.rochester.edu/research/synchronization/

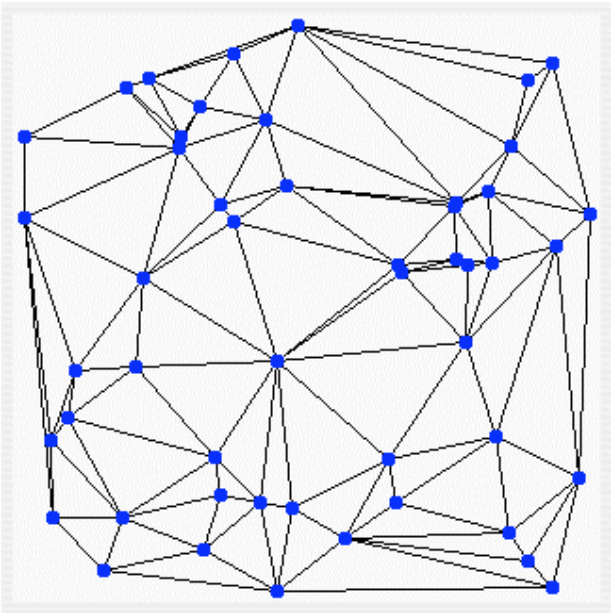
PODC, August 2007

Transactional Memory

- Alternative to locks for shared-memory synchronization
- User labels code fragments as atomic; system executes concurrently whenever possible
- Subject of great current interest—both language semantics and software/hardware support
- Conspicuous lack of driving applications: "chicken-and-egg" problem

Delaunay Triangulation

- Given 2-D set of points, find triangulation s.t. no circum-circle contains another point
- Several important properties & applications; may be *refined* w/extra pts to eliminate narrow triangles [Kulkarni et al., 2006]
- Best sequential algorithm [Dwyer] is O(n loglog n); we leverage this



Benchmark Implementation

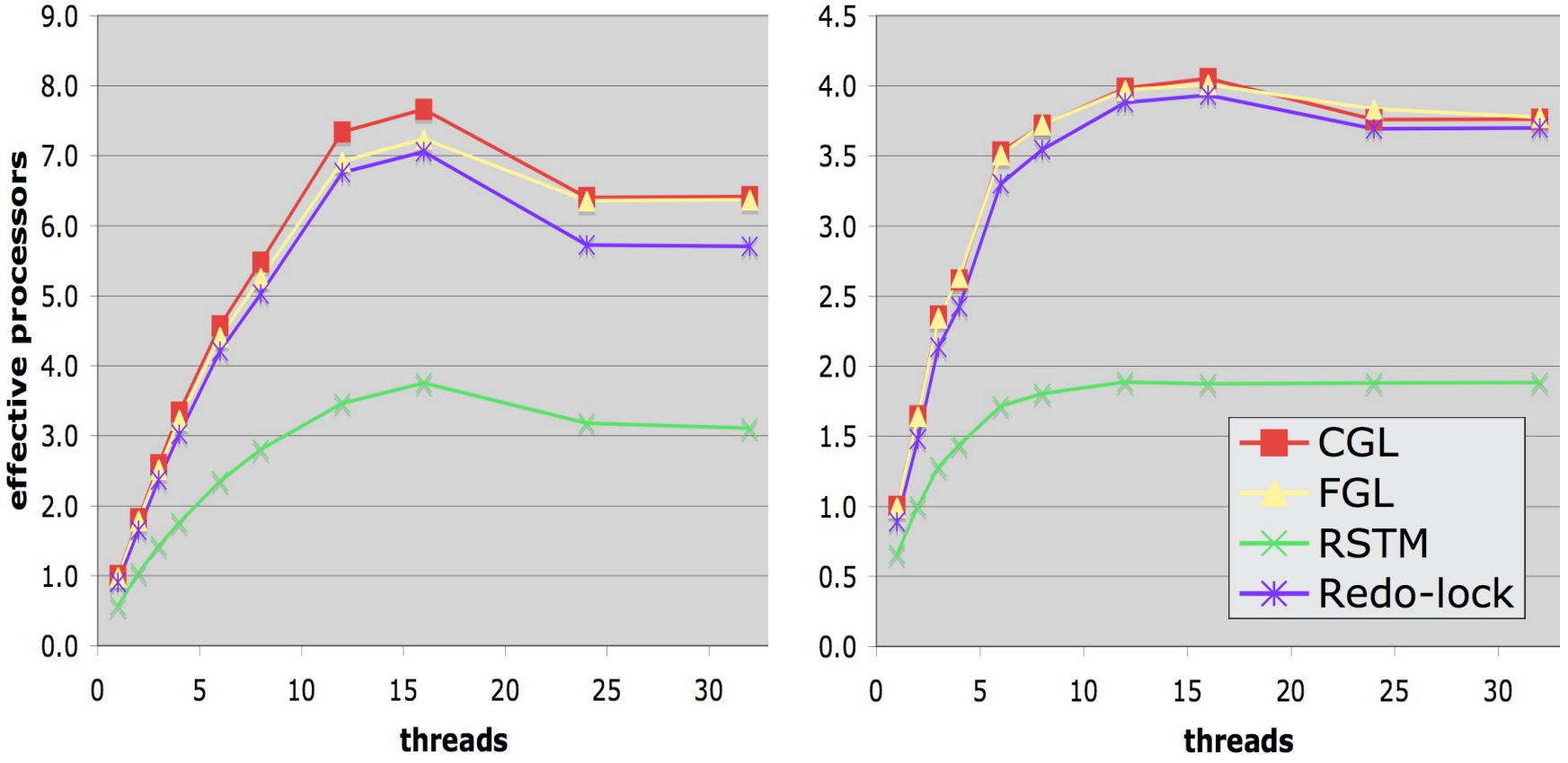
- Written in C++ using the RSTM2 API (see paper at TRANSACT on Thursday)
- 3200 lines; 24 source files
- 3 transactional object types
  - edge, edge_ref, list_node
- 3 static transactions (~440 lines — 14%)

- Alternating transactional & *private* phases
- Private time dominates (~95%)

Performance Notes

- Speedup is memory-limited
- Similar performance for coarse and fine-grain locks, indirection-free STM
  - » no performance benefit for FGL or STM
  - » but nice to be safe up-front
- 2X performance hit for indirection, incurred by private computation

Speedup (200K points)



16p SunFire
4.1s

sequential time

8c Niagara
9s

Programming Notes

- RSTM2 API facilitated experiments
  - "smart pointers" for reasonable syntax, efficient access hooks, static error checks
  - generics for transactional / private sharing
  - wide variety of back-end implementations
- Nowhere near simple enough for naive users
  - issues with both convenience and deep semantics
  - see paper at TRANSACT on Thursday

Conclusions

- Barriers + transactions = useful programming idiom.
- Private code must not pay for the existence of TM!
- Library-based TM can be a productive vehicle for systems experimentation.
- TM for the masses will require language integration.

- See papers on the mesh application [IISWC '07], API experience [TRANSACT '07], and privatization [UR TR 915, following BA]
- Download our code!

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

www.cs.rochester.edu/research/synchronization/