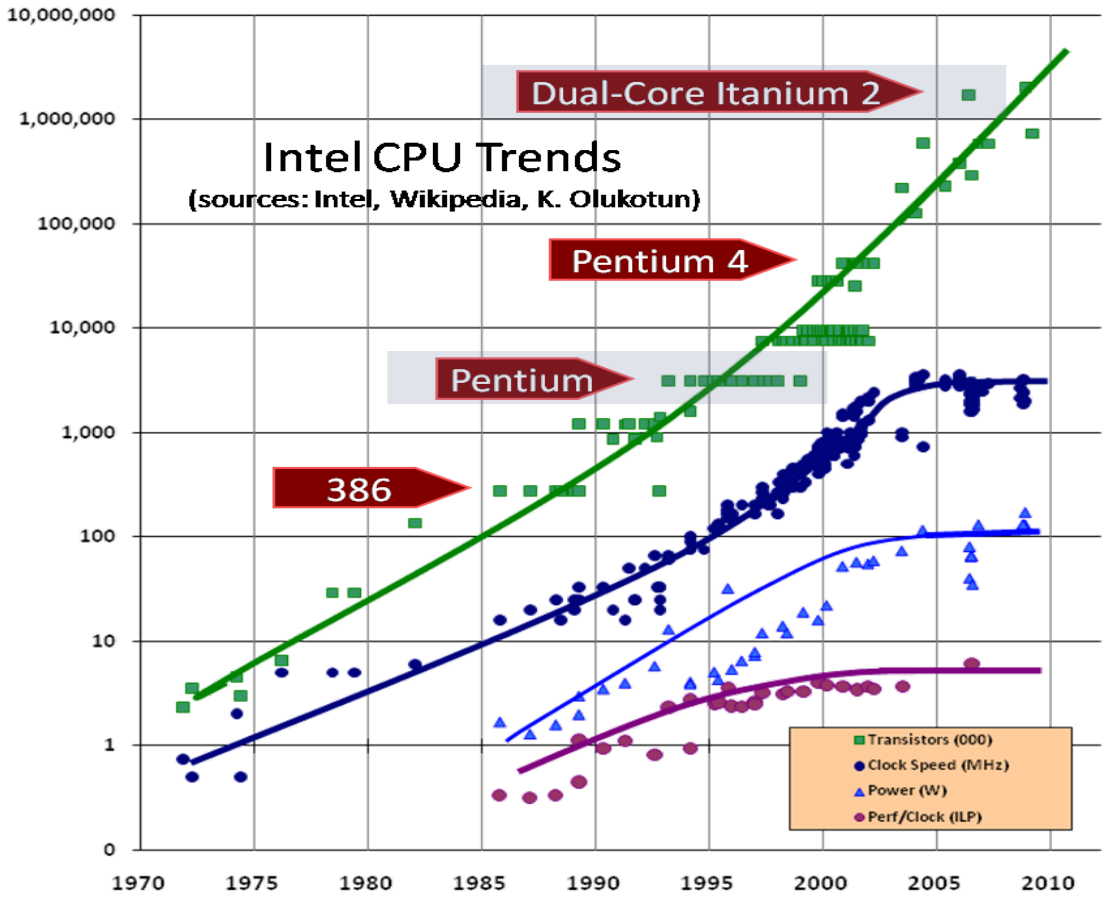
Parallel Programming
&

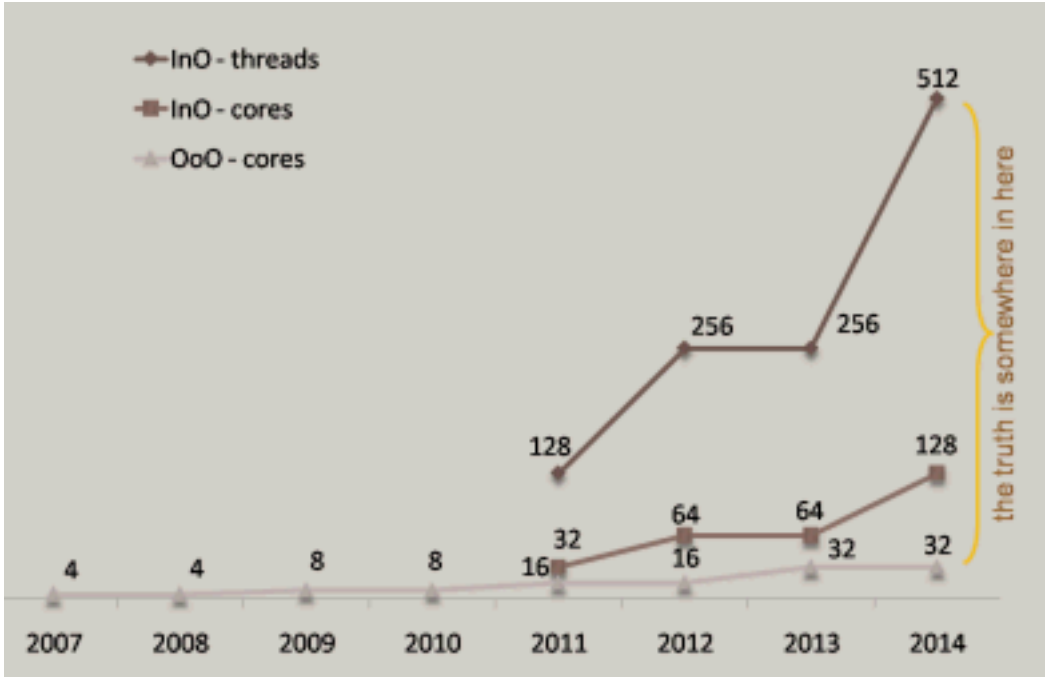

Murat Keçeli

Why do we need it now?



http://www.gotw.ca/publications/concurrency-ddj.htm

Why do we need it now?

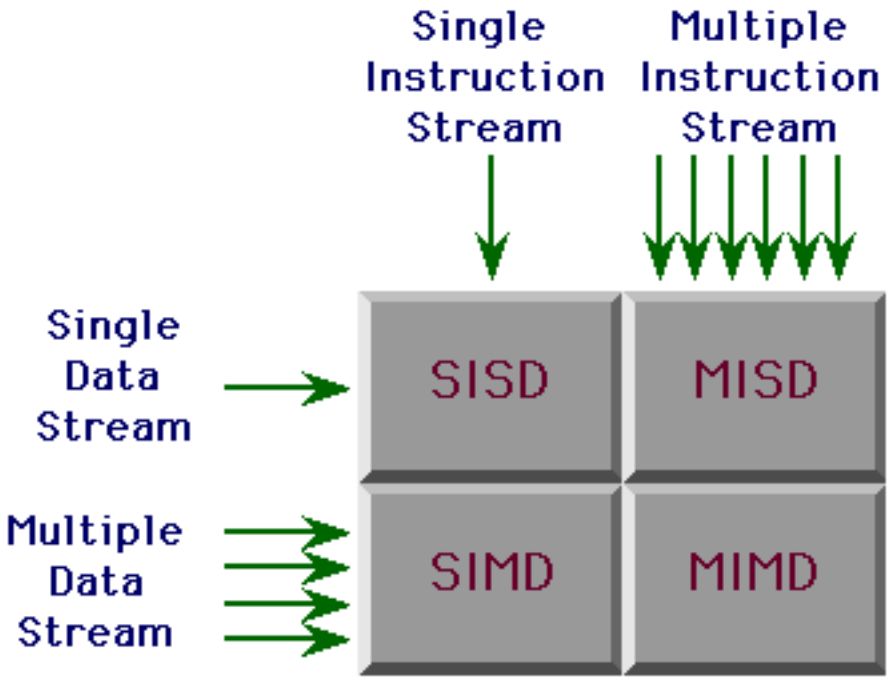


Intel® Xeon Phi™ Coprocessor 7120X
(16GB, 1.238 GHz, 61 core)

http://herbsutter.com/2012/11/30/256-cores-by-2013/

Flyyn's Taxonomy (1966)

Computer architectures



http://users.cis.fiu.edu/~prabakar/cda4101/Common/notes/lecture03.html

Multiple instruction multiple data

- Shared memory
  - All processors are connected to a "globally available" memory.
  - Your laptop, smartphone, a single node in a cluster.
  - Easier to implement, but not scalable.
- Distributed memory
  - Each processor has its own individual memory location.
  - Single processors at different nodes.
  - Data is shared through messages. Harder to implement.
- Hybrid (clusters, grid computing)
- Distributed shared memory (Distributed Global Address Space)

Grid and Cloud Computing

- Scalable solutions for loosely coupled jobs.
- Cloud is the evolved version of grid computing. (in terms of efficiency, QoS, reliability)
- Crowd-sourcing: SETI@HOME, FOLDIT@HOME,
- The clean energy project. 2.3 million organic compounds screened by volunteers to discover the next generation of solar cell materials. (World Community Grid, IBM)
- We can write proposals for thermochemistry calculations for aromatic hydrocarbons.

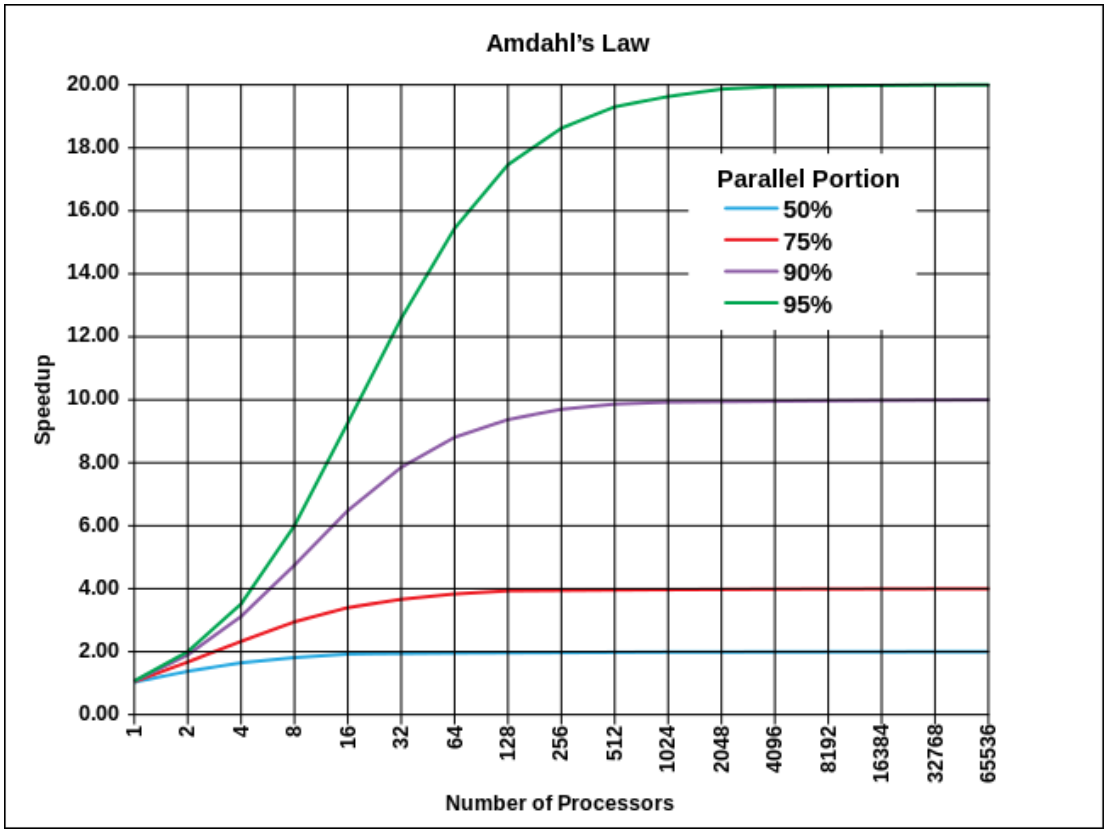
Goals of parallel programming

- Linear speedup: problem of a given size is solved N times faster on N processors
  - You can reduce time/cost

$$\text{Speedup} = \frac{\text{Serial execution time}}{\text{Parallel execution time}} \qquad S_N = \frac{t_1}{t_N} = N \qquad 0 < E_N = \frac{S_N}{N} \leq 1$$

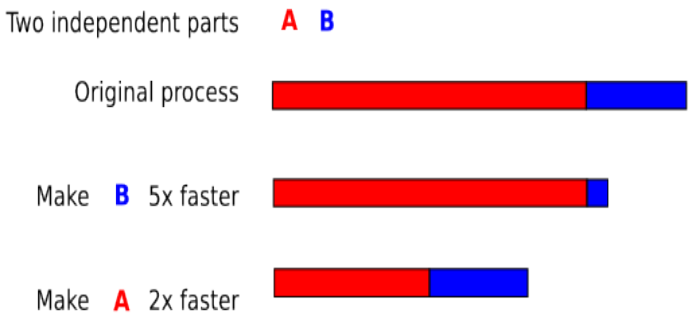
- Scalability: problem that is N times bigger is solved in the same amount of time on N processors
  - You can attack larger problems

Amdahl's law



$0 \le p \le 1$: parallel portion

$$S_N = \frac{1}{(1-p) + \frac{p}{N}}$$



http://en.wikipedia.org/wiki/Amdahl's_law

Parallelization Tools

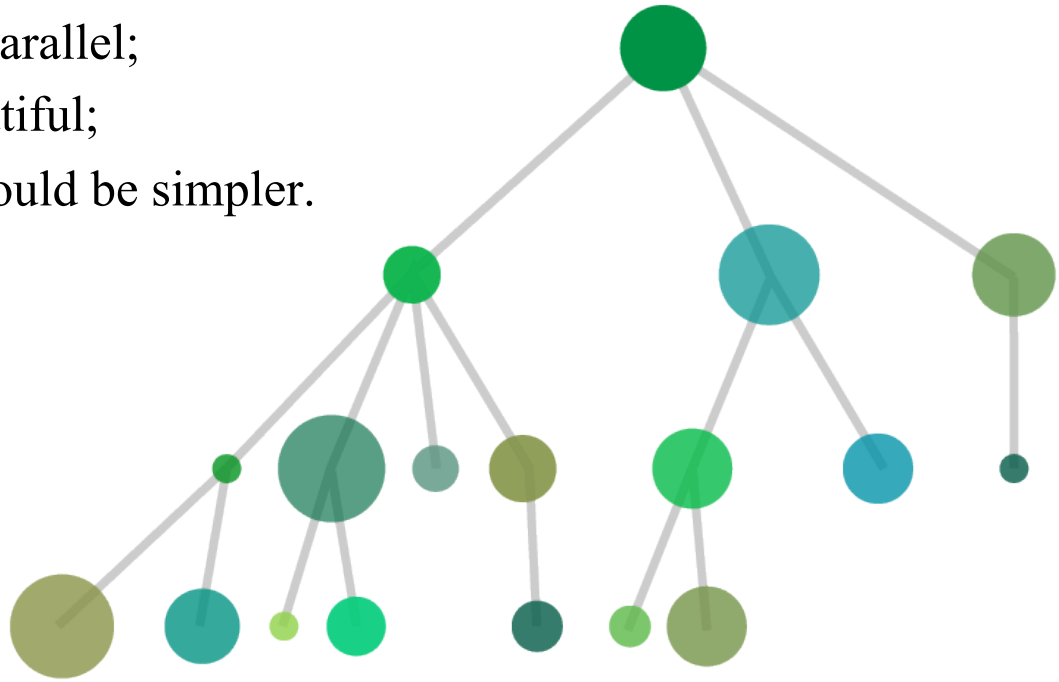
- Auto-parallelization
- Libraries (Intel Threading Building Blocks, Intel MKL, Boost)
- Cilk, Unified Parallel C, Coarray Fortran
- Functional programming languages (Lisp, F#)
- OpenMP (Open Multi-Processing, shared memory)
- MPI (Message Passing Interface, distributed memory),
- Java is designed for thread level parallelism, java.util.concurrent
- Python (https://wiki.python.org/moin/ParallelProcessing)
  - Global interpreter lock: The mechanism to assure that only one thread executes Python bytecode at a time

How to do parallel programming

- Start with the chunk that takes most amount of time.
- Decide the parallelization scheme based on available hardware and software.
- Divide the chunk into subtasks such that:
  - Minimum dependency (minimizes communication)
    - Each process has its own data (data independence)
    - Each process do not need others' functions to finish (functional independence)
  - Equal distribution (minimizes latency)
    - Workload is equally distributed

SCOOP

- Scalable COncurrent Operations in Python: is a distributed task module allowing concurrent parallel programming on various environments, from heterogeneous grids to supercomputers.
  - The future is parallel;
  - Simple is beautiful;
  - Parallelism should be simpler.



Hello World

```
1  from __future__ import print_function
2  from scoop import futures
3
4  def helloWorld(value):
5      return "Hello World from Future #{0}".format(value)
6
7  if __name__ == "__main__":
8      returnValues = list(futures.map(helloWorld, range(16)))
9      print("\n".join(returnValues))
```

- Results of a map are always ordered even if their computation was made asynchronously on multiple computers.

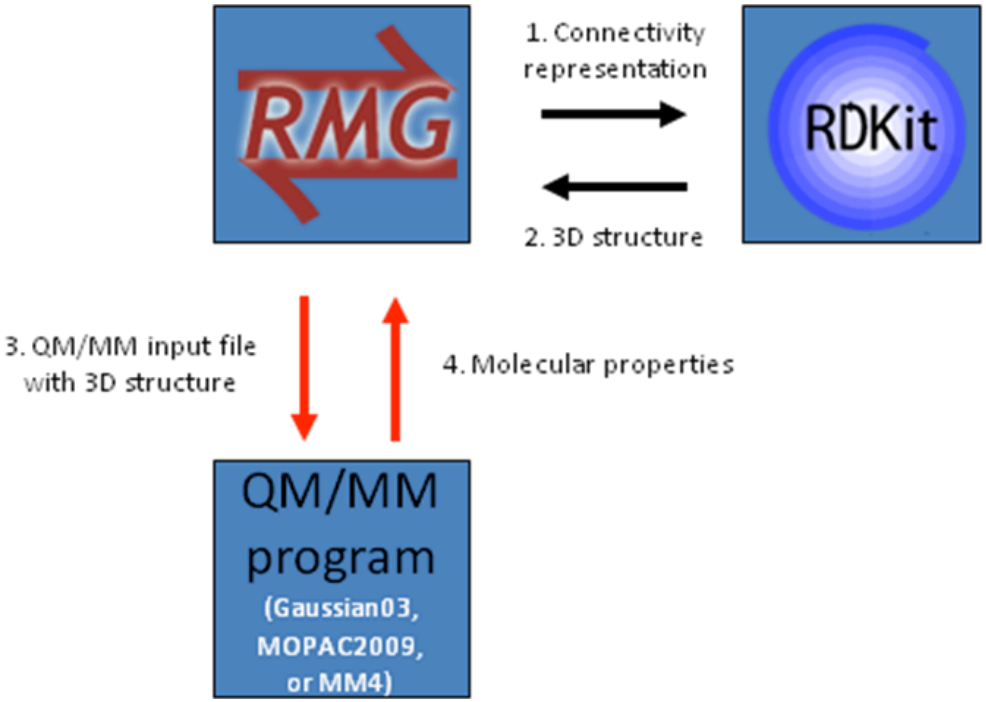
RMG & Thermochemistry

- Thermochemical parameters (enthalpy, entropy, heat capacity) are important for reaction equilibrium constants, kinetic parameter estimates, and thermal effects.
- Affects both the mechanism generation process and the behavior of the final resulting model.
- **Estimate** based on the group additivity approach of Benson.
  - This method is fast and can be improved by adding more parameterization.
  - Harder to parallelize: Hierarchical search, database sharing
  - Currently fails for aromatic species and subject to fail for any species outside of its parametrization scope.
  - As the applications of RMG starts to vary, this module needs to be updated for *ad hoc* corrections.

QMTP (Greg Magoon)

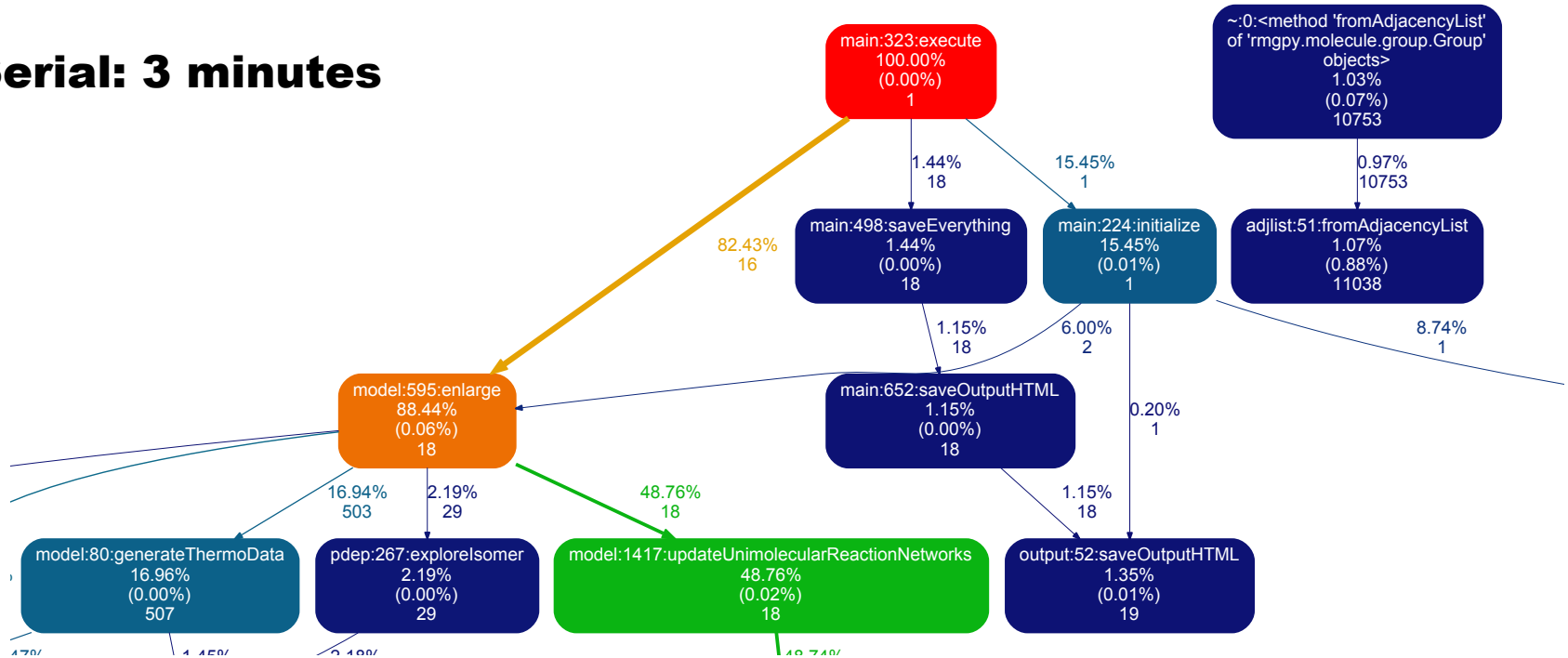
- Quantum mechanics thermodynamic property (QMTP) module is designed for on-the-fly quantum and force field calculations **to calculate** thermochemical parameters.
  - Must be linked to third party programs.
  - Error checking is required.
  - Slow. Speed depends on the method of calculation and the software chosen.
  - Calculations are uncoupled. (embarrassingly parallel) Much easier to parallelize.
  - Both speed and reliability improvement comes from outside.

QMTP Design


RMG
1. Connectivity representation
RDKit
2. 3D structure
3. QM/MM input file with 3D structure
4. Molecular properties
QM/MM program
(Gaussian03, MOPAC2009, or MM4)

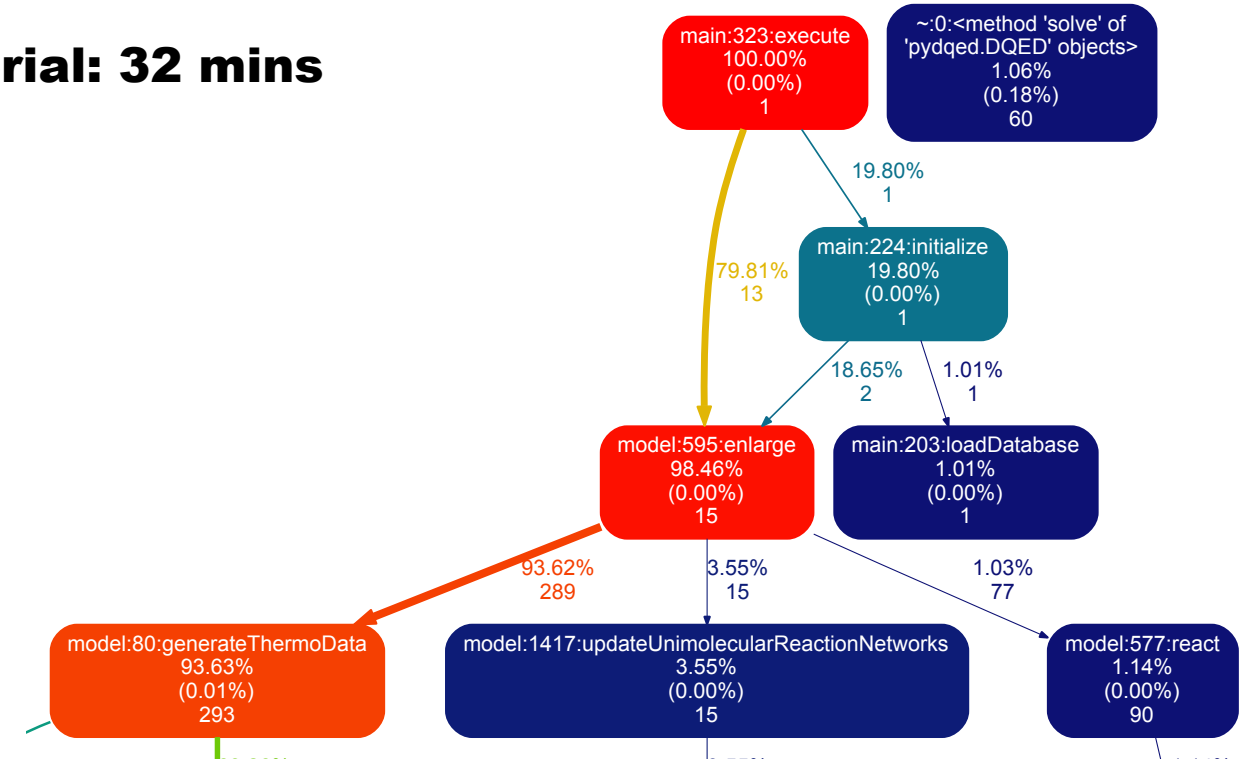
1,3-Hexadiene without QM

Serial: 3 minutes



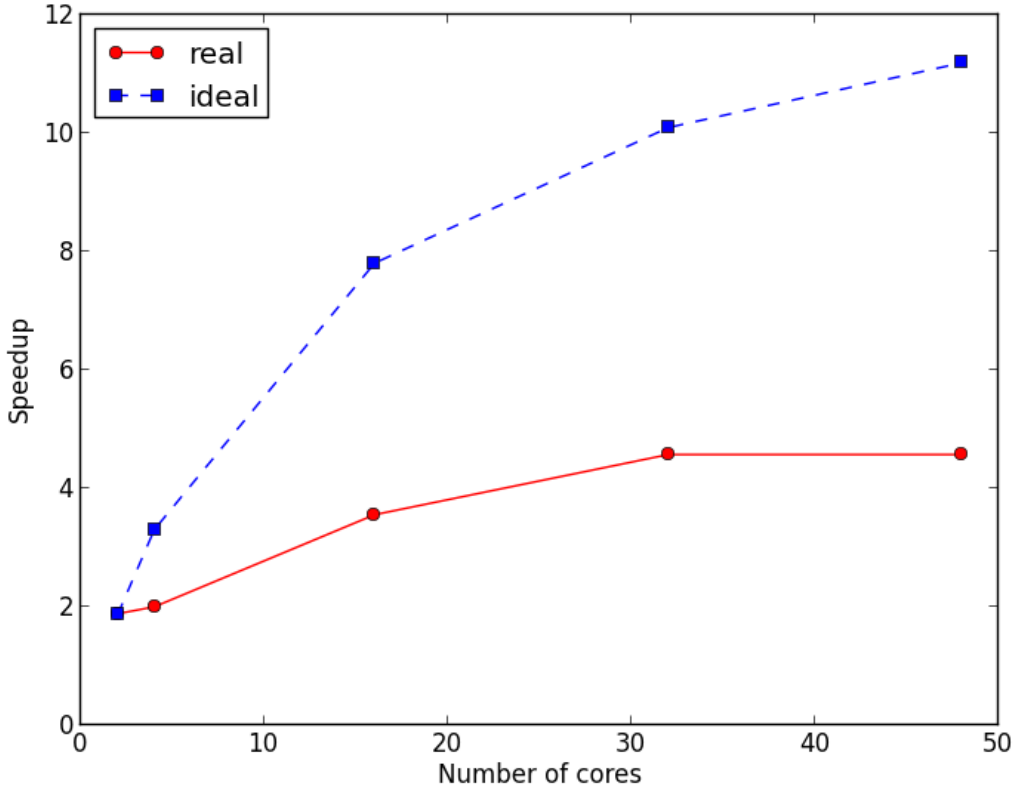
1,3-Hexadiene with Mopac PM3

Serial: 32 mins



Current situation

36 cores: 7 mins



Problem-1

- Job submission through grid engine fails.
  - ImportError: libRDGeneral.so.1: cannot open shared object file: No such file or directory
- More info @ https://groups.google.com/forum/#!topic/scoop-users/T7bXN5x1zic
  - "SCOOP won't be handling environment variables directly (at least the way as MPI does). The next version (0.7) will contain a new feature called a prolog which is an executable (ie. a shell script) that SCOOP will execute at the launch of every worker. Exporting environment variables will be possible in this prolog."
- There might be a trick. (.bashrc, .login is not the solution)
- Links required can be installed by root.
- Temporary solution:
  - Submit a sleep job, ssh to that node, and run interactively. Don't forget to cancel them. (kill -9 -1, then qdel xxxx)

Problem-2

- Job fails *sometimes*.
  - AttributeError: 'ccData' object has no attribute 'rotcons'
  - ERROR:root:Not all of the required keywords for success were found in the output file!
- Reason: Unpredictable buffering of I/O by OS.
- Trying: os.fsync, os.path.getsize
- Temporary solution: Add a sleep part in the code. (1 second seems ok for MOPAC jobs, Gaussian jobs are more tricky)

Conclusions

- Parallel was the future, now we all need it.
- Redesigning some portions of RMG-Py is necessary.
  - Reactor conditions, Pdep calculations, graph search
  - Database structure (Tree, might not be the best option)
  - Minimize I/O.
  - Avoid writing to home disk. Move them when job finishes.
  - We can avoid sharing library with workers.
- Parallel programming is a headache even for the most advanced programmers.
- You may think that you solve some problems by sleeping, but it is only a dream, it won't last long.

Thank you all

RMG-Dev

Northeastern University

References

- http://web0.tc.cornell.edu/Services/Education/Topics/Parallel/
- https://computing.llnl.gov/tutorials/parallel_comp/
- http://www.ibm.com/developerworks/library/wa-cloudgrid/
- http://parajava.sourceforge.net/
- http://groups.csail.mit.edu/cag/ps3/
- http://www.intel-software-academic-program.com/courses/

Monte-Carlo computation of pi

- Generate a random point inside unit square
  - Two random numbers $0<x,y<1$
- The probability of having this point inside the quarter of unit disc is pi/4