Uniform Distributions on Integer Matrices

The Problem

- How many ways can I fill in a matrix with specified row and column sums?

?	?	?	2
?	?	?	2
?	?	?	3
2	2	3	

A Quick Example

?	?	?	2
?	?	?	2
?	?	?	3
2	2	3	

0	2	0	2
1	0	1	2
1	0	2	3
2	2	3	

Who Cares?

- Theoretical motivation:
  - Computer scientists are interested in optimal algorithms for enumeration – this is a test bed
- Practical applications:
  - Anything that’s a network or might be made to look like one

The Social Network

- I have a bunch of friends and you have a bunch of friends and our friends have a bunch of friends and so on

- This can be represented as a binary matrix

The Social Network

- Suppose we see some patterns (structures) in this matrix

- Question:
  - Are the structures simply a result of the fact that some people have more friends than others?
  - Or is something else going on?

The Social Network

- Solution:
  - Fix the row and column sums (how many friends people have)
  - Sample uniformly from the matrices satisfying these sums
  - See whether the structures of interest are likely to occur by chance

Darwin’s Finches

- Biologists are also interested in these sorts of matrices
- Though no longer obviously network problems
- For example: Darwin observed some a number of species of finches distributed across a series of islands

Darwin’s Finches

- Question:
  - Is the distribution of finches explained by the fact that some islands are bigger and can support more species than others and some finches are more common than others?
  - Or are competitive dynamics at work?

Darwin’s Finches

- Solution:
  - Test by uniformly sampling with fixed number of species per island (column sums) and fixed number of islands on which a species is found (row sums)
  - Can test how likely observed combinations of species are to occur by chance

Computational Challenge

- A small example…
- How many 8x8 matrices with row & column sums = 8 do you think are there?

Computational Challenge

- Answer:
  - 1,046,591,482,728,408,076,517,376
  - (a.k.a. 1 septillion plus)

Computational Challenge

- As many as that is, this is whittled down from more than 40^64 possibilities

- Which is > 10^100 or about 10 times the age of the universe

- Even though our algorithm doesn’t actually require us to sort though all these possibilities, we do have to handle the fact that many many (sub)problems lead to the same place
  - … parallel sort & removal of duplicates = lots of headaches

Strategies: Pros & Cons

- Level-by-level vs. depth first + hashing
  - rank synchronization
- Storage: distributed vs. master-worker model
- Sorting:
  - Merge sort
  - Radix sort
- Pre-computing coefficients?

Our Algorithm

- Serial algorithm
  - multimaps, new maps, and more
  - Class to hold “Problems”
  - Not the most interesting from a parallel computing standpoint, but several days of coding…

Our Algorithm

- Parallel step by step
  - Rank 0 gets initial problem, calculates random matrix used in load balancing for sort, and broadcasts
  - For each row, each rank goes though:
    - Step 0. Calculate new maps for my problems
    - Step 1. Package the new problems
    - Step 2. Send and receive all the info (6 all to all calls)
    - Step 3a. Unpack the received data from all ranks
    - Step 3b. Merge sort & deduplicate problems received
    - Step 4. Convert to array which stores final answer (portion of DAG) for this row
  - Print final DAG & total solutions to file

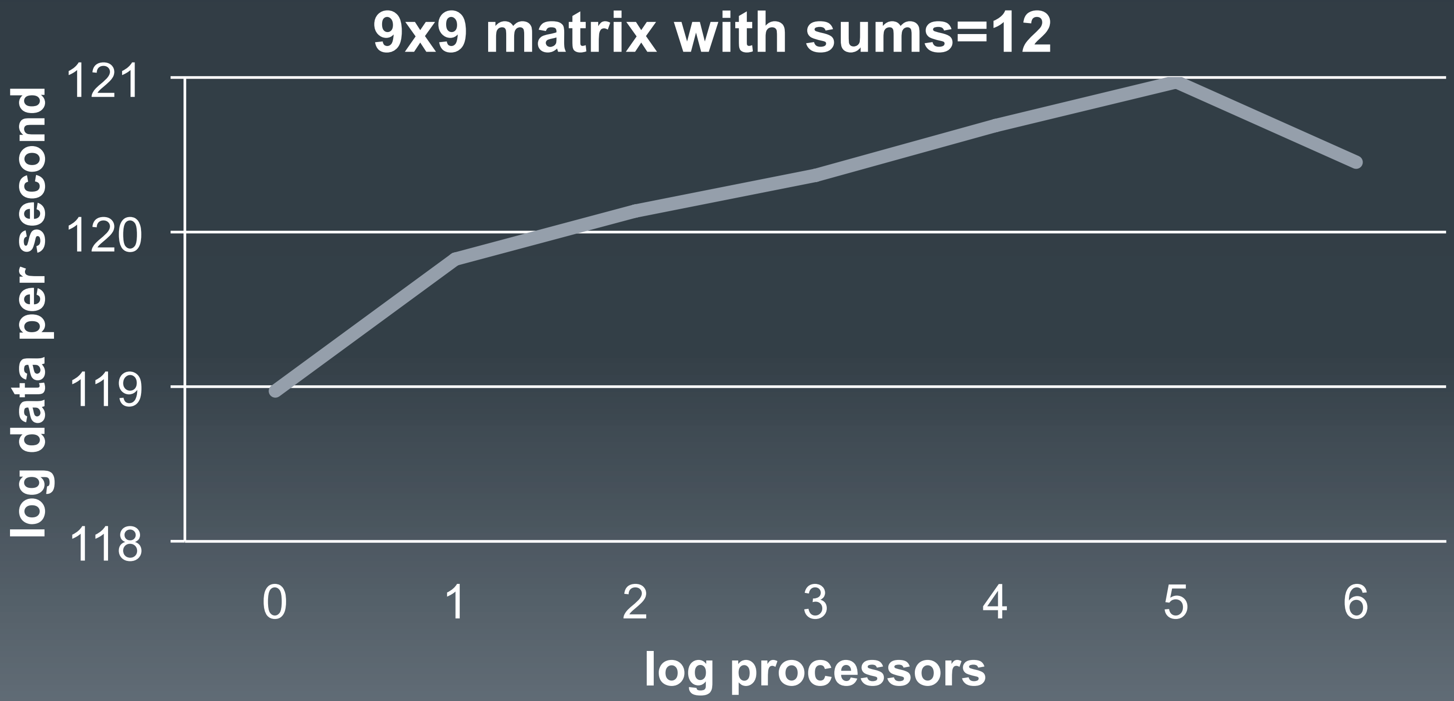
Did we get the right answer?

- Multiple checks of the final answer …
  - Against small problems we could compute by hand
  - Against serial code written by Jeff Miller in Python

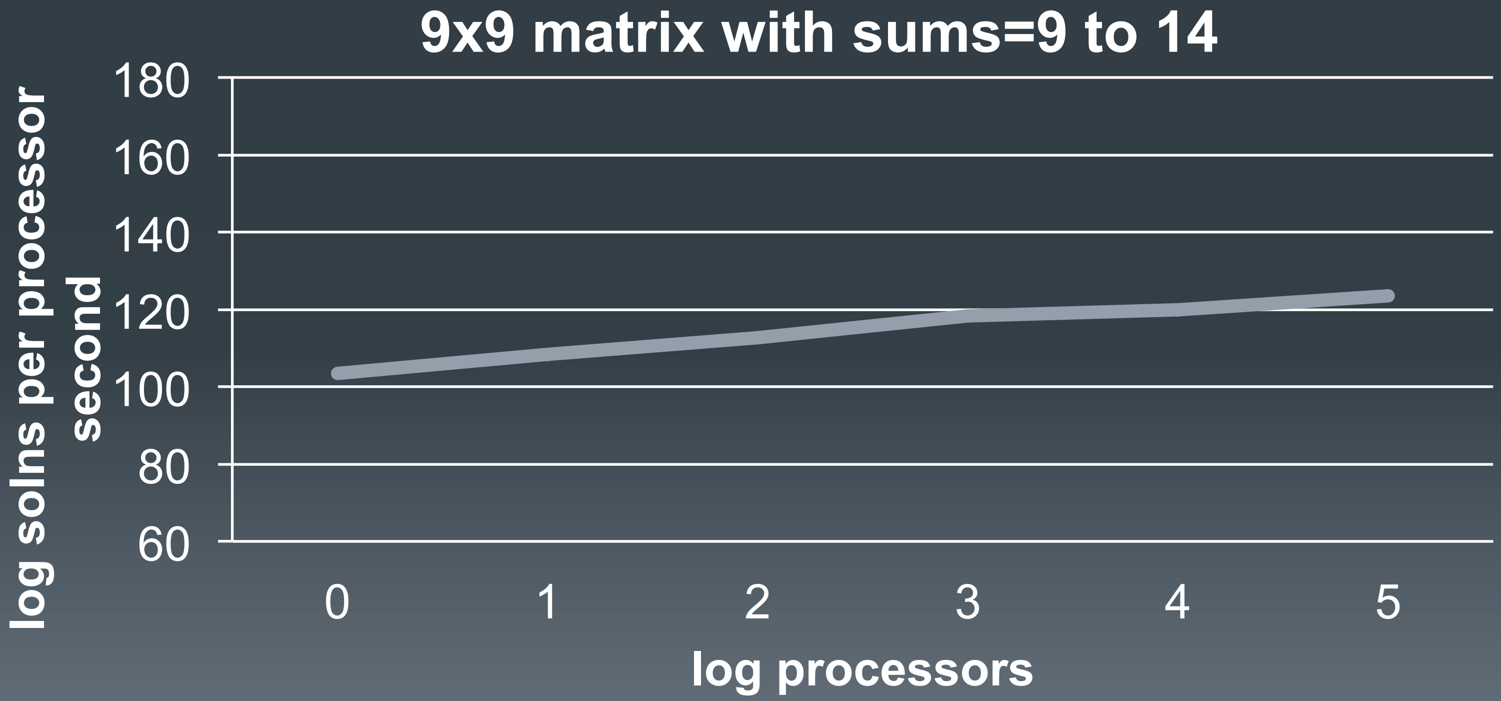
- And lots of printouts & testing individual function performance

* Though note answers are floating point … if we really wanted correct down to the last decimal place would need some sort of arbitrary precision implementation

Strong Scaling



Weak Scaling(?)



Future Directions

- Improve parallel sort
  - Better randomization & load balancing

- Improve serial algorithm
  - Can we do a better job of anticipating duplicates?

- Deal with really big numbers (arbitrary precision arithmetic)

- Start using for real sampling problems!