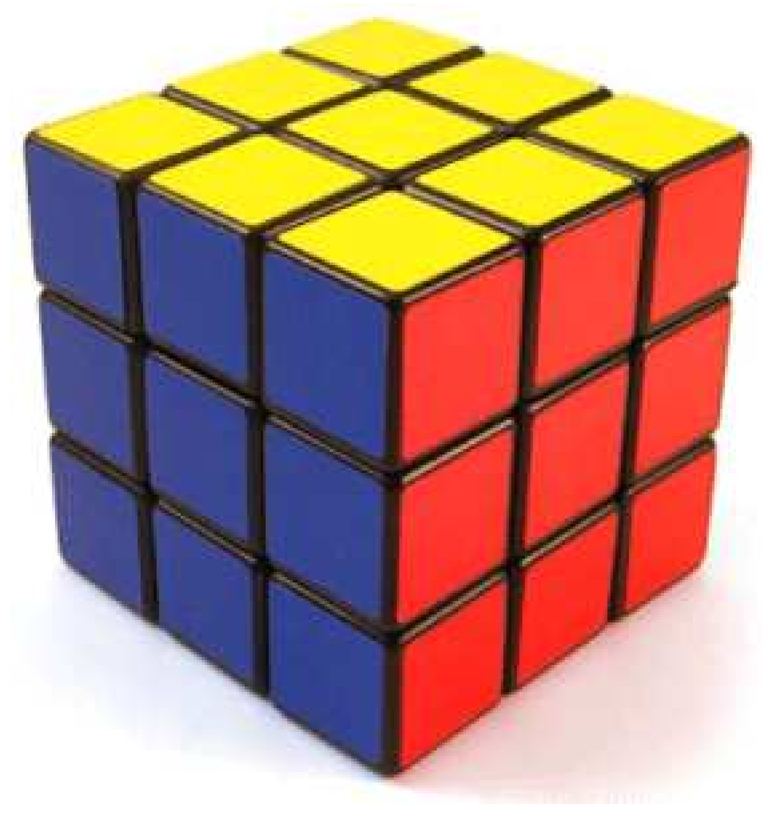
Using Parallel Disk-Based Computing: a New Record for Computer-Generated Solutions to Rubik's Cube

Gene Cooperman
(Joint work with Daniel Kunkle)



History of Rubik's Cube

- Invented in late 1970s in Hungary.
- In 1982, in *Cubik Math*, Singmaster and Frey conjectured:

> *No one knows how many moves would be needed for "God's Algorithm" assuming he always used the fewest moves required to restore the cube. It has been proven that some patterns must exist that require at least seventeen moves to restore but no one knows what those patterns may be. Experienced group theorists have conjectured that the smallest number of moves which would be sufficient to restore any scrambled pattern — that is, the number of moves required for "God's Algorithm" — is probably in the low twenties.*

- Current Best Guess: 20 moves suffice
  - States needing 20 moves are known

History of Rubik’s Cube (cont.)

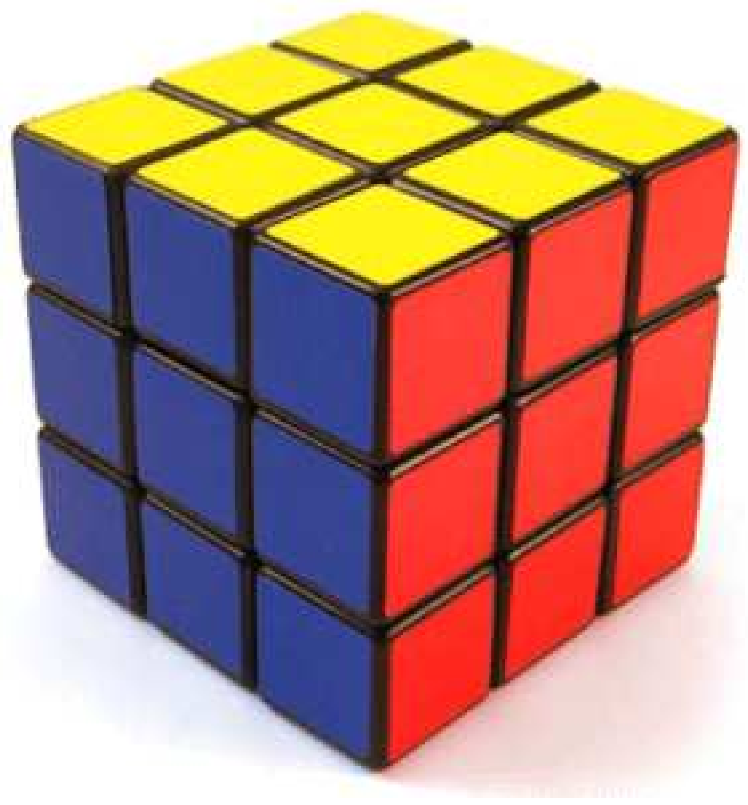
- Invented in late 1970s in Hungary.
- 1982: “God’s Number” (number of moves needed) was known by authors of conjecture to be between 17 and 52.
- 1990: C., Finkelstein, and Sarawagi showed 11 moves suffice for Rubik’s $2 \times 2 \times 2$ cube (corner cubies only)
- 1995: Reid showed 29 moves suffice (lower bound of 20 already known)
- 2006: Radu showed 27 moves suffice
- 2007 Kunkle and C. showed 26 moves suffice (and computation is still proceeding)
- D. Kunkle and G. Cooperman, “Twenty-Size Moves Suffice for Rubik’s Cube”, *International Symposium on Symbolic and Algebraic Computation* (ISSAC-07), 2007, ACM Press, pp. 235–242.

Solution of Rubik's Cube: Humans

- 4.3×10^{19} states
- Solutions by human beings
  1. Solve one face (Now only 1.7×10^{9} states remain.)
  2. Memorize move sequences that preserve that first face.
  3. Use those sequences to solve second face,
     while preserving first face.
  4. REPEAT for third face (while preserving first two faces), etc.

Notation

- Generators: Up (U), Down (D), Front (F), Back (B), Left (L), Right (R)
- Reachable states of cube: cube = $\langle U, D, F, B, R, L \rangle$
- Number of states: $|cube| = 4.3 \times 10^{19}$

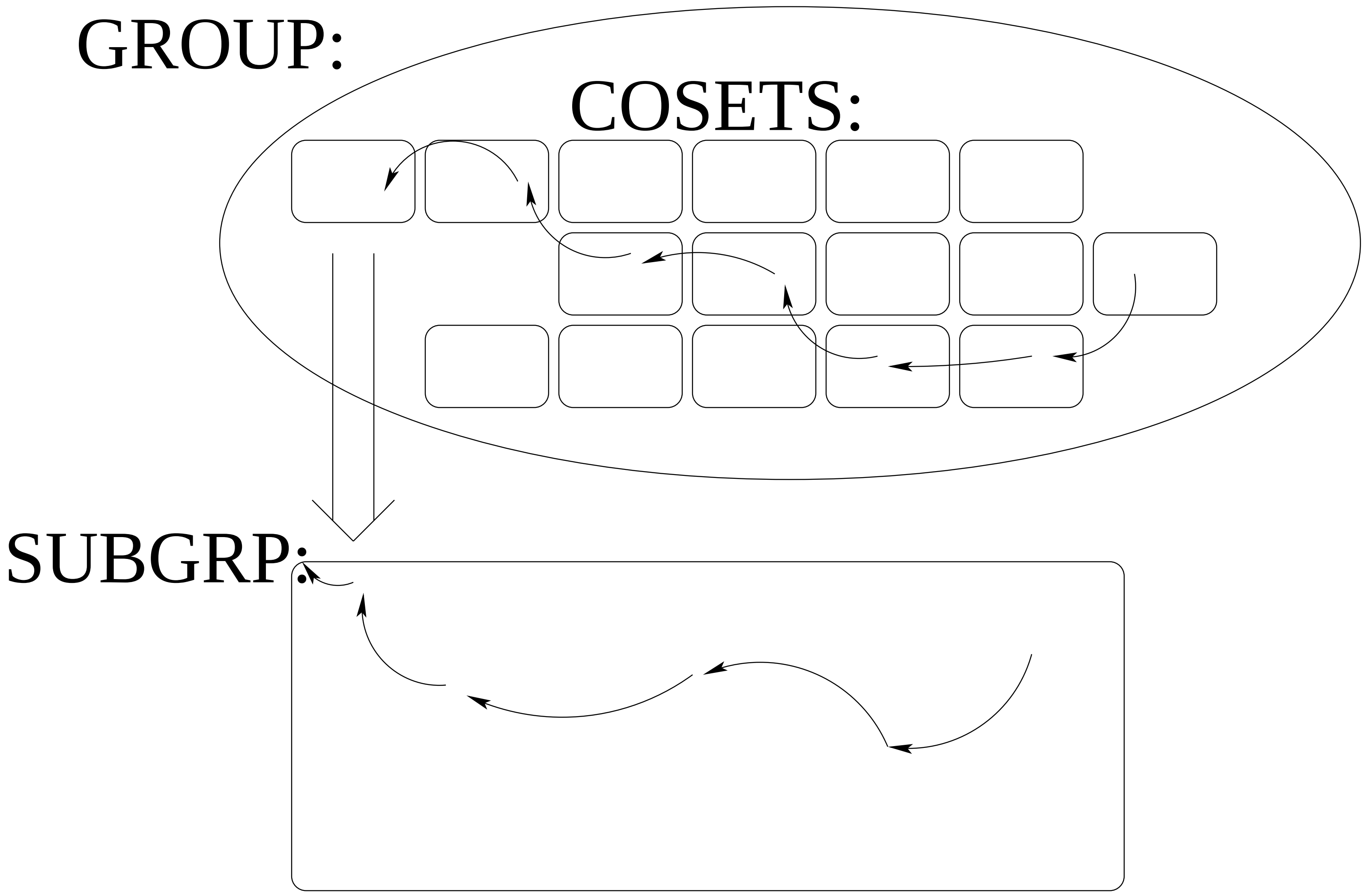


Solution of Rubik’s Cube by Computer: 1995

- $|cube| = 4.3 \times 10^{19}$ states
- Consider subgroup $S = \langle U, D, L^2, R^2, F^2, B^2 \rangle$
- $|cube|/|S| = 2.2 \times 10^9$; $|S| = 2.0 \times 10^{10}$

1. $|cube|/|S|$: Use shortest possible sequence of moves in Rubik’s cube so that remaining confi guration is reachable via generators: U, D, L^2, R^2, F^2, B^2
2. $|S|$: Starting from confi guration in $S = \langle U, D, L^2, R^2, F^2, B^2 \rangle$, make shortest possible moves to solve it.

Solution of Rubik's Cube: Cosets in Mathematical Group Theory



Optimization: Use Symmetries of Geometric Cube: 1995

- Optimization: Use up to 48 symmetries of the geometric cube. Note that these symmetries take generators (U, D, L, R, F, B) to generators.
- S preserves 16 of the 48 symmetries of a geometric cube. So, only have to solve problem for:
  - $|cube|/|S| = (2.2/16) \times 10^9$; $|S| = (2.0/16) \times 10^{10}$
  - $|cube|/|S| = 1.3 \times 10^8$; $|S| = 1.2 \times 10^9$
  - **Only a billion cases (1.2×10^9) to check!!**

Solution of Rubik's Cube by Computer (cont.): 1995

- $|cube| = 4.3 \times 10^{19}$ states
- Consider subgroup $S = \langle U, D, L^2, R^2, F^2, B^2 \rangle$
- $|cube|/|S| = 2.2 \times 10^9$; $|S| = 2 \times 10^{10}$

1. $|cube|/|S|$: 12 possible moves
2. $|S|$: 18 possible moves
3. 1995 (Reid): Total moves: 12 + 18 = 30 moves suffice
4. 1995 (Reid): Only a few cases needing 12 moves in $|cube|/|S|$; Solve them individually: 11 + 18 = 29 moves suffices
5. 2006 (Radu): 27 moves suffice
   Show by direct solution that some smaller cases in $|cube|/|S|$ can be solved directly.
   11 + 18 - 2 = 27 moves suffice

Solution of Rubik's Cube by Computer: 2007

- $|cube|$ = 4.3×10^{19} states
- Consider *square subgroup* $Q = \langle U^2, D^2, L^2, R^2, F^2, B^2 \rangle$
- $|cube|/|Q| = 6.5 \times 10^{13}$; $|Q| = 6.6 \times 10^5$

1. $|cube|/|Q|$: Use shortest possible sequence of moves in Rubik's cube so that remaining confi guration is reachable via generators: U^2, D^2, L^2, R^2, F^2, B^2
2. $|Q|$: Starting from confi guration in $Q = \langle U^2, D^2, L^2, R^2, F^2, B^2 \rangle$, make shortest possible moves to solve it.

Optimization: Use Symmetries of Geometric Cube: 2007

- Optimization: Use up to 48 symmetries of the geometric cube. Note that these symmetries take generators (U, D, L, R, F, B) to generators.
- Q preserves all 48 symmetries of a geometric cube. So, only have to solve problem for:
  - $|cube|/|Q| = (6.5/48) \times 10^{13}$; $|Q| = (6.6/48) \times 10^5$
  - $|cube|/|Q| = 1.4 \times 10^{12}$; $|Q| = 1.4 \times 10^4$
  - **Only a trillion cases (1.4×10^{12}) to check!!**

Solution of Rubik's Cube by Computer (cont.): 2007

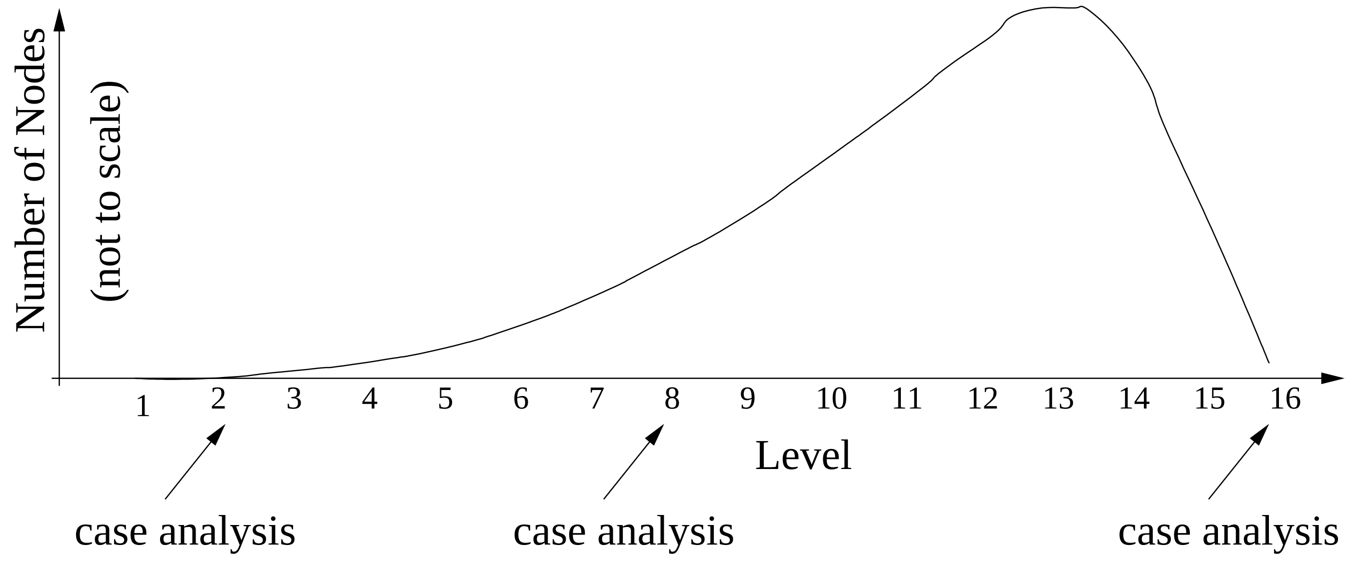
- $|cube| = 4.3 \times 10^{19}$ states
- Consider subgroup $Q = \langle U^2, D^2, L^2, R^2, F^2, B^2 \rangle$
- $|cube|/|Q| = 6.5 \times 10^{13}$; $|Q| = 6.6 \times 10^5$

1. $|cube|/|Q|$: 16 possible moves
2. $|Q|$: 13 possible moves
3. Kunkle and Cooperman: Total moves: 16 + 13 = 29 suffice
4. 2007: Only a few cases needing 16 moves in $|cube|/|Q|$; Solve them individually: 15 + 13 = 28 moves suffices
5. 2007: 26 moves suffice
   Show by direct solution that some smaller cases in $|cube|/|Q|$ can be solved directly using 2 fewer moves.
   15 + 13 - 2 = 26 moves suffice

$|cube|/|Q|$: symmetry classes of cosets of square subgroup

Level	Elements	Level	Elements		Level	Elements	
0	1	6	38336	$\approx 3.8 \times 10^4$	12	140352357299	$\approx 1.4 \times 10^{11}$
1	1	7	490879	$\approx 4.9 \times 10^5$	13	781415318341	$\approx 7.8 \times 10^{11}$
2	3	8	6298864	$\approx 6.3 \times 10^6$	14	421980213679	$\approx 4.2 \times 10^{11}$
3	23	9	80741117	$\approx 8.1 \times 10^7$	15	330036864	$\approx 3.3 \times 10^8$
4	241	10	1028869318	$\approx 1.0 \times 10^9$	16	17	
5	3002	11	12787176355	$\approx 1.3 \times 10^{10}$			
		Total	1357981544340	$\approx 1.36 \times 10^{12}$			

Table 1: Distribution of symmetrized cosets of the square subgroup.



Summary

Subgroup S		Subgroup Q (square subgroup)	
Largest search: 2.0×10^{10} after symmetries: $(2.0/16) \times 10^{10} = 1.2 \times 10^9$		Largest search: 6.5×10^{13} after symmetries: $(6.5/48) \times 10^{13} = 1.4 \times 10^{12}$	
Year	Moves Needed	Year	Moves Needed
1995	12+18 = 30	2007	16+13 = 29
1995	11+18 = 29	2007	15+13 = 28
2006	11+18-2 = 27	2007	15+13-2 = 26

- Time using square subgroup (Q):
  1. 63 cluster hours (16 8-way nodes) to show 16+13 = 29
     (in a parallel computation using TOP-C)
  2. Many hours on sequential machines to reduce to 15+13-2 = 26
- Further reductions? …

Two Primary Techniques Used

1. Fast multiplication of symmetrized cosets ($>$ 10,000,000 multiplies per second)
2. Use of large amounts of intermediate disk space (7 TB) for hash array (for duplicate elimination)

Fast Multiplication: $>$ 10,000,000 mults/second

- Table-based multiplication; Form smaller subgroups, factor each group element into the smaller subgroups; Use tables for fast multiplication among the small subgroups
- Tables are kept mostly in L1 cache; Most subgroups have less than 100 elements; Multiplication table has $< (100)^2$ elements, or $< 10,000$.
- Group of Rubik's cube
  - Group of permutations acting only on corner cubies
  - Group of permutations acting only on edge cubies
    - ∗ Flips of the two faces of each edge cubies (while holding location of edge cubie fi xed)
    - ∗ Moving edge cubies (while ignoring flips of the two faces)

Fast Multiplication (cont.)

- Moving edge cubies (while ignoring flips of the two faces)
  - Moving edge cubies using half-twists (180 degrees) only:
    - $*$ Half-twists split the 12 edge cubies into three invariant subsets, each containing 4 edge cubies (can't move edge cubie from one subset to the other using only half-twists)
  - Moving edge cubies using quarter-twists (but "divided by" half-twists: using the group theory concept of cosets and normal subgroups)

LONGER-TERM GOALS

- Why did we do it?

  1. Because it's there? (Yes, but . . .)
     - State space search occurs across a huge number of scientific disciplines. A popular challenge provides a crossroads where different disciplines can compare the power of their methods on a common ground.
  2. Because the world is running out of RAM!
     - A commodity motherboard holds only 4 GB RAM.
     - We now have 4- and 8-core motherboards, but no one will be putting eight times as much RAM on a *commodity* motherboard.

LONGER-TERM GOALS (cont.)

- The world is changing, as we near the end of Moore's Law.
  - Memory chips are no longer twice as dense every 18 months.
  - Large RAM is still available on server-class motherboards.
  - But the commodity market doesn't want to pay that premium.
  - So, those of us doing large scientifi c computations are being left out in the cold. We still need those ever larger memories – especially as the trend toward multi-core CPUs places ever more pressure on RAM.
- Our solution is to use disk as the new RAM! (See next slide.)

Disk-Based Parallel Computing

Disk is the New RAM

- Bandwidth of Disk: ˜ 100 MB/s
- Bandwidth of 50 Disks: 50 $\times$ 100 MB/s = 5 GB/s
- Bandwidth of RAM: approximately 5 GB/s

- Conclusion:

1. **CLAIM:** A computer cluster of 50 quad-core nodes, each with 200 GB of idle disk space, is a good approximation to a shared memory computer with 200 CPU cores and a *single* subsystem with 10 TB of shared memory.
   (The arguments also work for a SAN with multiple access nodes, but we consider local disks for simplicity.)
2. The disks of a cluster can serve as if they were RAM. (See the next slides for the issue of disk latency.)
3. The traditional RAM can then serve as if it were cache.

What are the issues in treating a cluster

. . . as a 10 TB shared memory computer?

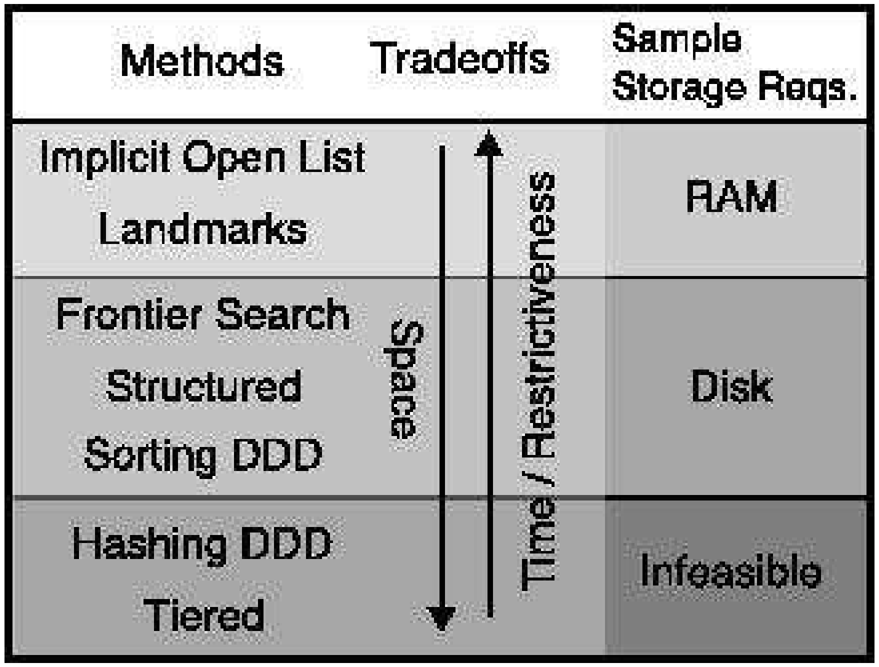
1. We require a *parallel* program. (We must access the local disks of many cluster nodes in parallel.)
2. The latency problem of disk.
3. Can the network keep up with the disk?

Three previous large disk-based computations in computational algebra had already been accomplished in joint work with E. Robinson and J. Müller. This gave us the confi dence to pose a general principle. Only latency will be discussed in this talk, but there is good reason to believe that the other two issues can also be overcome.

Overcoming Latency

- There are well-understood building blocks for using disk efficiency and for replacing the latency of disk by multiple streaming passes:
  - external sorting, B-trees, Bloom filters, Delayed Duplicate Detection, Distributed Hash Trees (DHT), and some still more exotic algorithms.

Space-Time Tradeoffs using Additional Disk



Methods	Tradeoffs	Sample Storage Reqs.
Implicit Open List Landmarks		RAM
Frontier Search Structured Sorting DDD	Space Time / Restrictiveness	Disk
Hashing DDD Tiered		Infeasible

“A Comparative Analysis of Parallel Disk-Based Methods for Enumerating Implicit Graphs”, Eric Robinson, Daniel Kunkle and Gene Cooperman, *Proc. of 2007 International Workshop on Parallel Symbolic and Algebraic Computation* (PASCO ’07), ACM Press, 2007, pp. 78–87

Rubik's Cube: Sorting Delayed Duplicate Detection

1. Breadth-fi rst search: storing new frontier (open list) on disk
2. Use Bucket Sorting to sort and eliminate duplicate states from the new frontier
   (The bucket size is chosen to fi t in RAM *(the new cache)*.
3. Storing the new frontier requires 6 terabytes of disk space (and we would use more if we had it). Saving a larg new frontier on disk prior to sorting *delays duplicate detection*, but makes the routine more effi cient due to economies of scale.

Rubik's Cube: Two-Bit trick

1. The final representation of the state space (1.4×10^{12} states) could use only 2 bits per state. (We use 4 bits per state for convenience.)
2. We used mathematical group theory to derive a highly dense, perfect hash function (no collisions) for the states of $|cube|/|S|$.
3. Our hash function represents *symmetrized cosets* (the union of all symmetric states of $|cube|/|S|$ under the symmetries of the cube).
4. Each hash slot need only store the level in the search tree *modulo 3*. This allows the algorithm to distinguish states from the current frontier, the next frontier, and the previous frontier (current level; current level plus one; and current level minus one). This is all that is needed.

QUESTIONS?