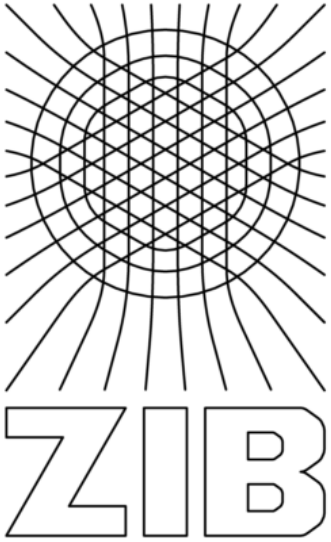
newthinking
store

O'REILLY®
www.oreilly.de



Solving Puzzles with MapReduce

Alexander Reinefeld and **Thorsten Schütt**
Zuse Institute Berlin

Hadoop Get Together, 29.09.2009, Berlin

How to solve *very* large combinatorial problems

… which cannot be solved on a single computer

… which do not fit into main memory

… which require

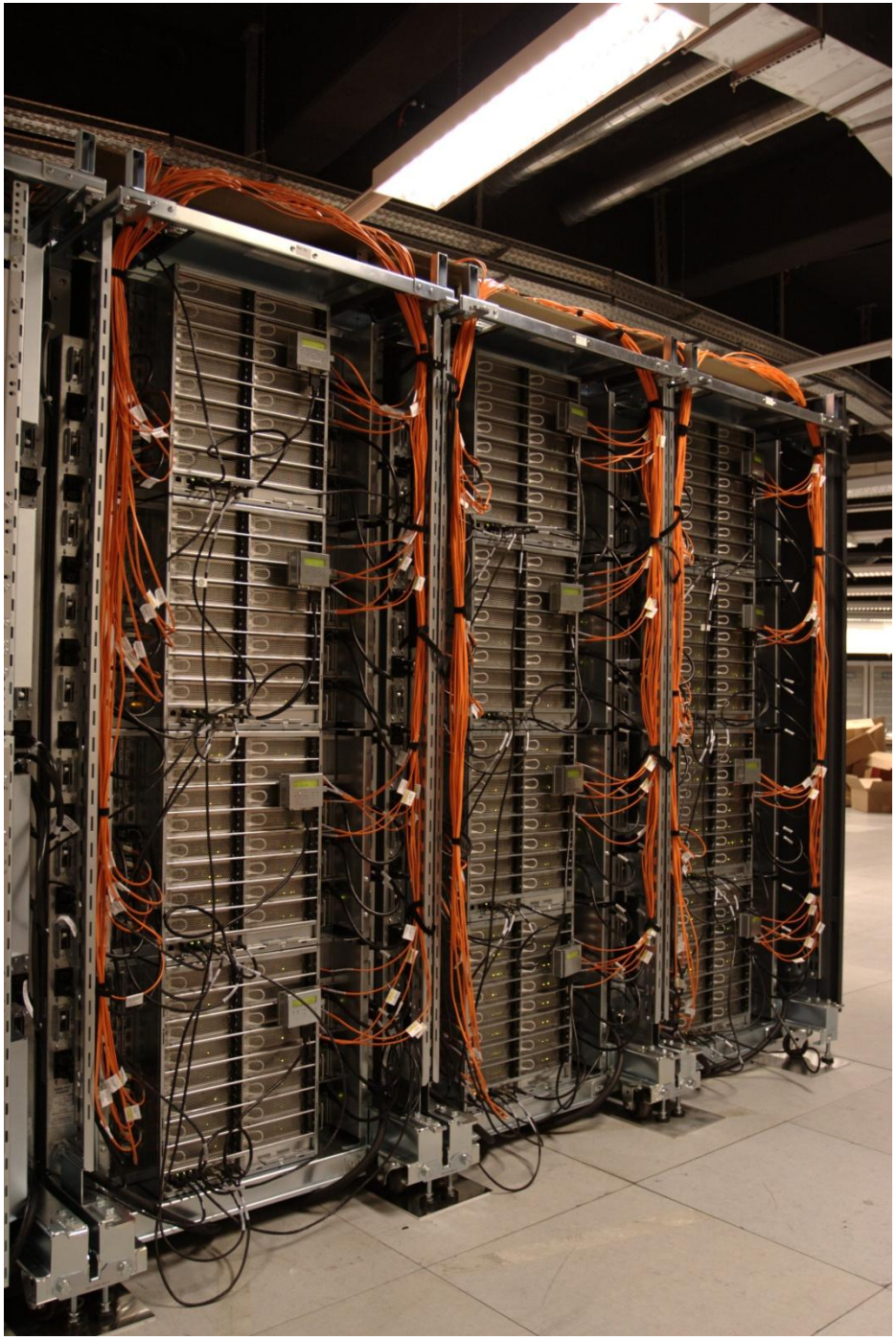
- massively parallel computers (or clusters)
- or distributed computers (cloud computing, datacenters)

15-Puzzle

16!/2 = 10,461,394,944,000 states

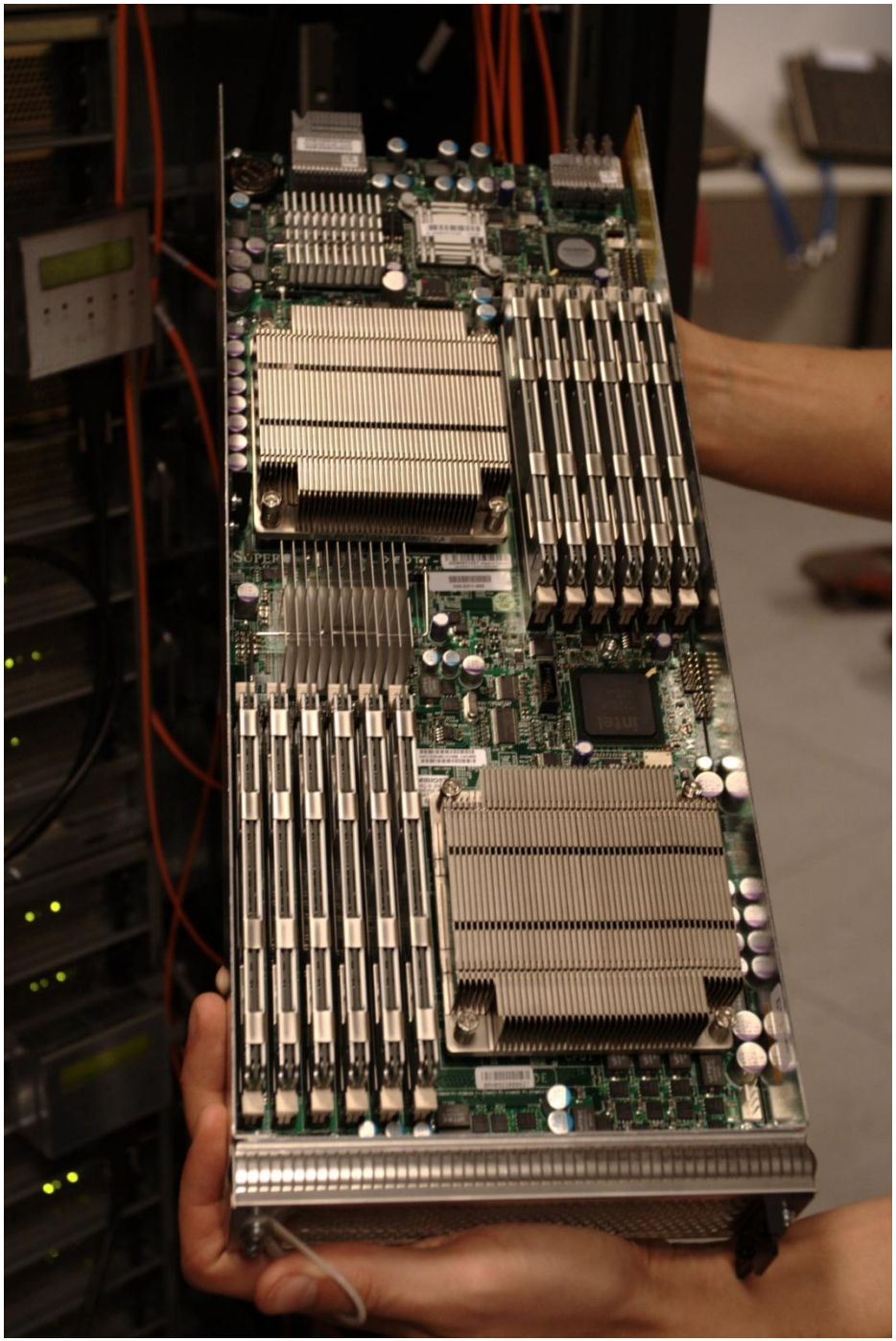
What is the hardest configuration?

Disclaimer:
Talk is about Map-Reduce!
Not about Hadoop!



- 1500 Compute Nodes
- Large Parallel File System

- No Local Disk!
- Just …
  - CPU
  - Memory
  - Infiniband (20/40 Gbit)



Hadoop

- No Local Disks
- Parallel File System (Lustre)
- Batch System

- Waste of Disk Space

- Was unstable (2 years ago)

Breadth-First Search

Breadth-First Search

```
main() {
   Queue queue = [start_node];                    // work-queue
   Set visited = {};                              // visited set
   while (queue != []) {
        foreach(Position pos in neighbors(queue.pop_first())) {
                if(pos not in visited) {          // check is visited?
                        visit(pos);               // visit
                        visited.add(pos);
                }
        }
   }
}
```

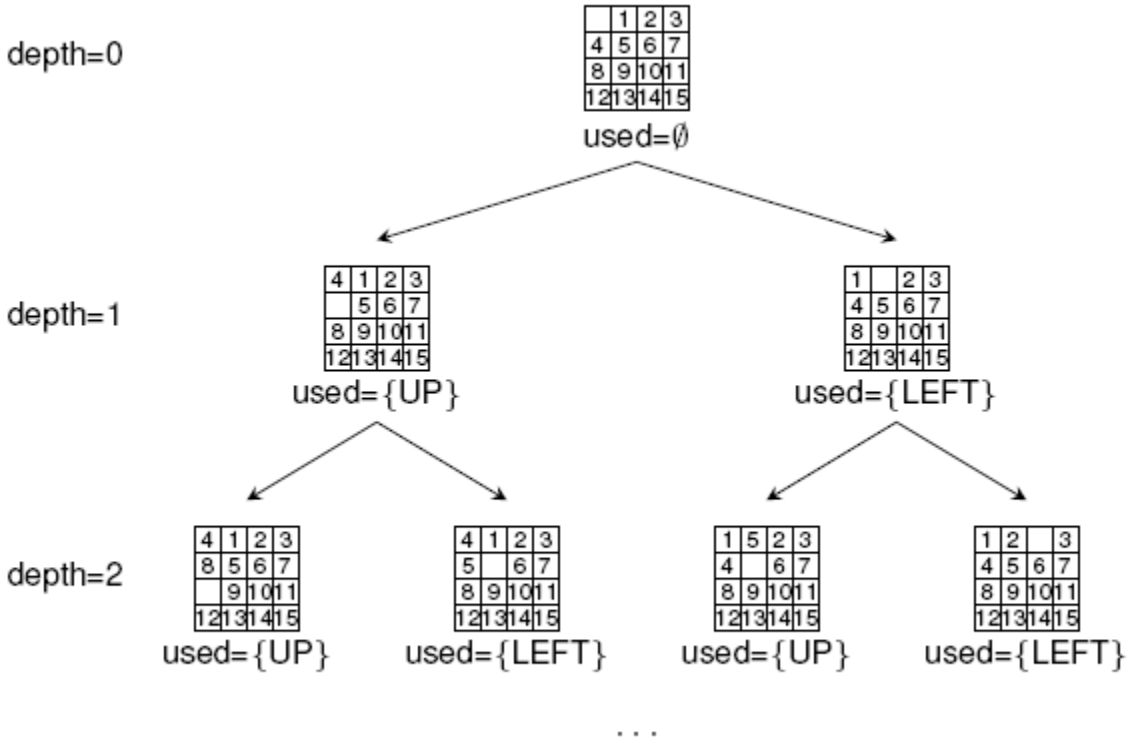
Breadth-First Search

```
main() {
    Queue queue = [start_node];                    // work-queue
    Set visited = {};                              // visited set
    while (queue != []) {
        foreach(Position pos in neighbors(queue.pop_first())) {
            if(pos not in visited) {               // check is visited?
                visit(pos);                        // visit
                visited.add(pos);
            }
        }
    }
}
```

Imagine `visited` has a size of **10,461,394,944,000 - 1**

Breadth-First Frontier Search

- Key: Position
- Value: Store „used moves“
  - U, D , L, R
  - 4 Bit



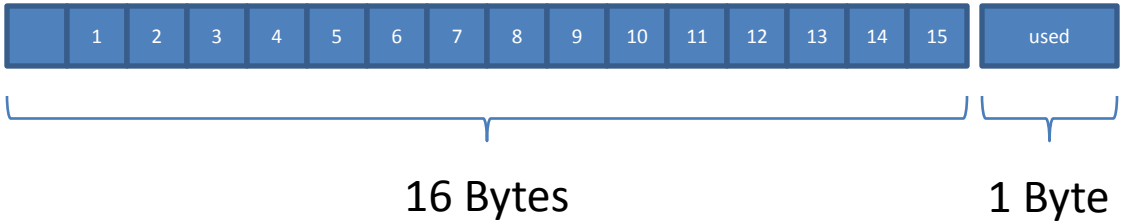
Key/value pairs in the 15-puzzle

- key = position
- value = move(s) that generated that pos.

Breadth-First Frontier Search with MapReduce

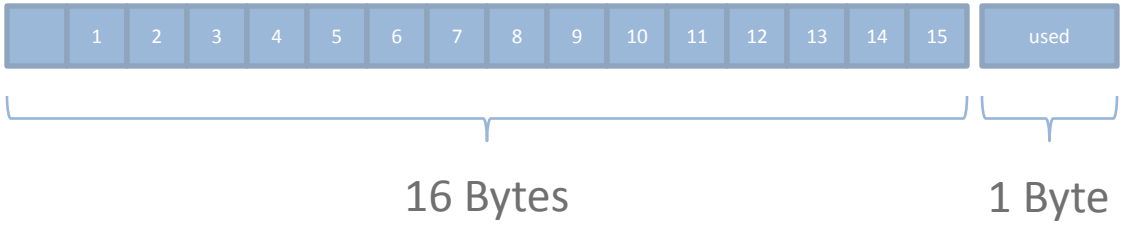
```
void mapper(Position position, set usedMoves) {
   foreach((successor, move) in successors(position))
      if(inverse(move) & usedMoves == mt)
         emit(successor, move);
}

void reducer(Position position, set<set> usedMoves) {
    moves := 0;
    foreach(move in usedMoves)
          moves := moves | move;
    emit(position, moves);
}

main() {
    front = [(start_node, 0)];
    while (front != mt) {
          intermediate = map(front, mapper());
          front = reduce(itermediate, reducer());
    }
}
```

How to store a Puzzle Position?

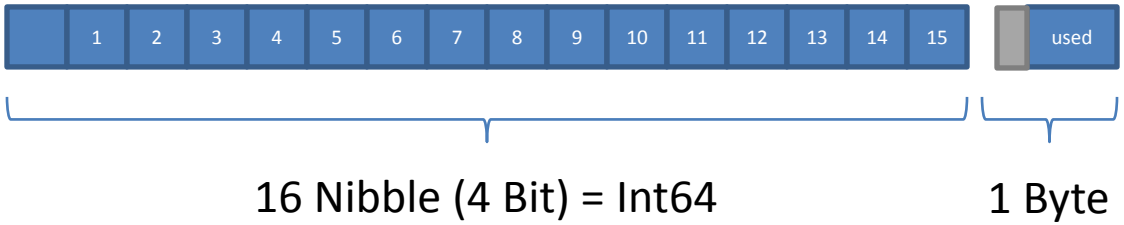


17 Bytes * 10,461,394,944,000
= 170TB

How to store a Puzzle Position?

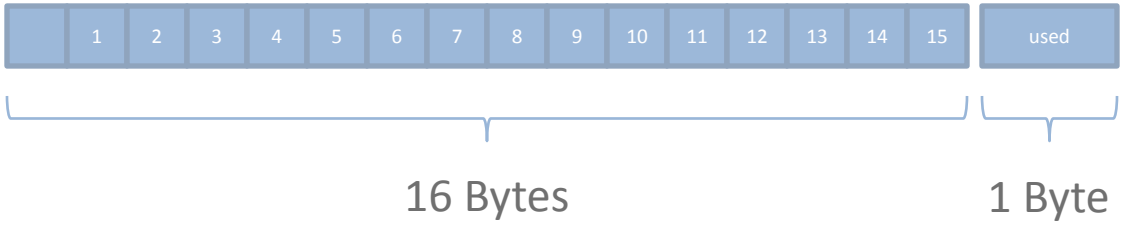


17 Bytes * 10,461,394,944,000
= 170TB

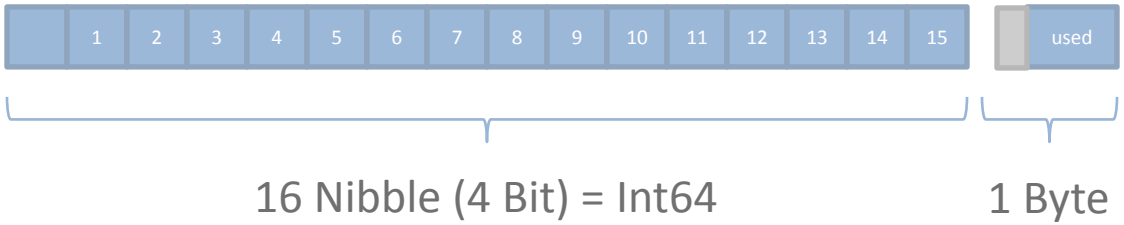


9 Bytes * 10,461,394,944,000
= 90TB
16 Bytes * 10,461,394,944,000
= 160 TB

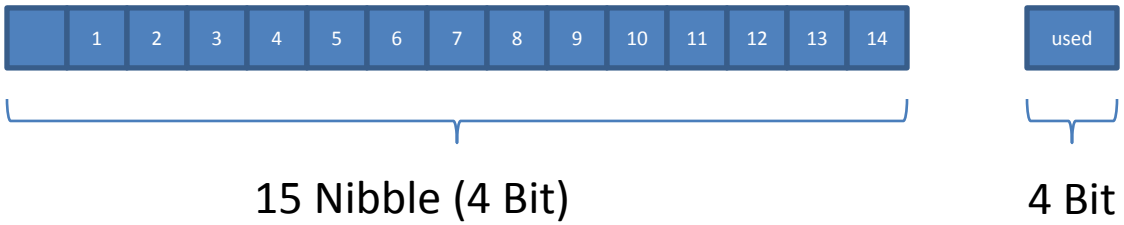
How to store a Puzzle Position?



17 Bytes * 10,461,394,944,000
= 170TB



9 Bytes * 10,461,394,944,000
= 90TB
16 Bytes * 10,461,394,944,000
= 160 TB



8 Bytes * 10,461,394,944,000
= 80TB

Int64 for Key + Value

66 h later ….

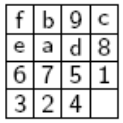
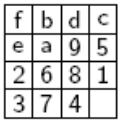
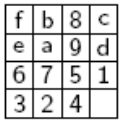
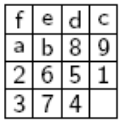
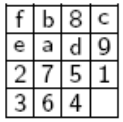
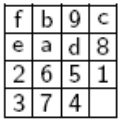
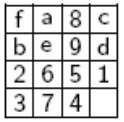
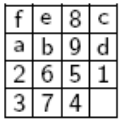
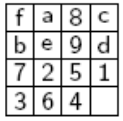
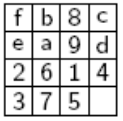
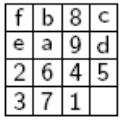
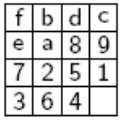
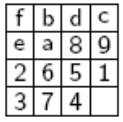
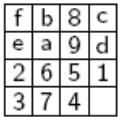
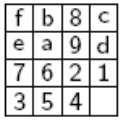
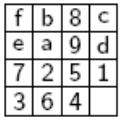
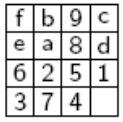
Result

16!/2 =
10,461,394,944,000 states

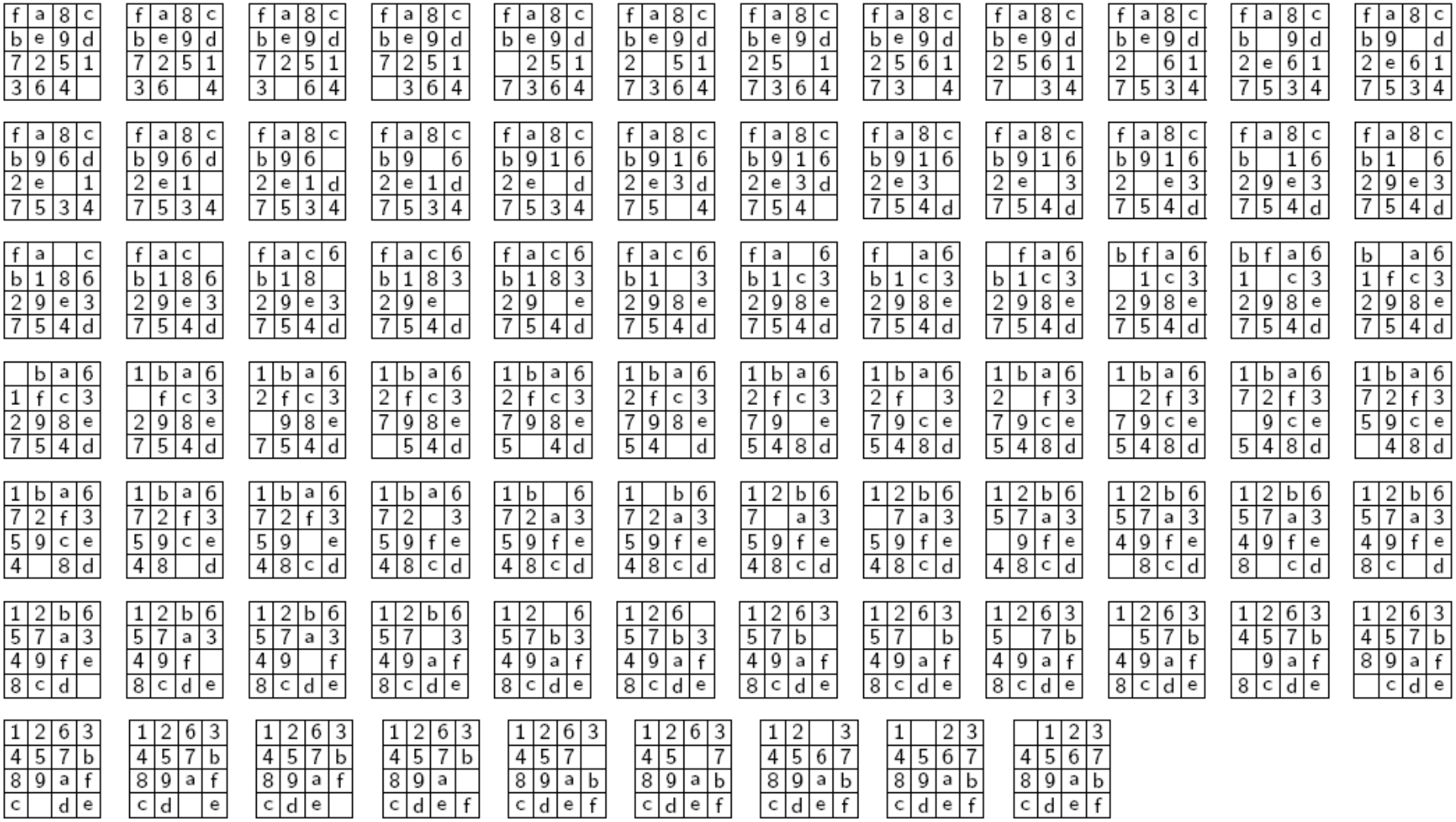
~66 hours execution time

d	map output	reduce output	d	map output	reduce output
0	0	1	41	105,232,854,740	83,099,401,368
1	2	2	42	148,081,981,988	115,516,106,664
2	4	4	43	203,644,257,664	156,935,291,234
3	10	10	44	273,637,877,374	208,207,973,510
4	24	24	45	358,812,754,412	269,527,755,972
5	54	54	46	458,895,663,682	340,163,141,928
6	108	107	47	571,810,278,100	418,170,132,006
7	214	212	48	693,606,328,846	500,252,508,256
8	454	446	49	818,215,994,990	581,813,416,256
9	966	946	50	937,634,709,510	657,076,739,307
10	2,012	1,948	51	1,042,659,315,592	719,872,287,190
11	4,060	3,938	52	1,123,608,988,412	763,865,196,269
12	8,122	7,808	53	1,171,775,867,988	784,195,801,886
13	16,156	15,544	54	1,180,827,850,290	777,302,007,562
14	32,338	30,821	55	1,171,775,867,988	742,946,121,222
15	63,794	60,842	56	1,073,945,899,830	683,025,093,505
16	125,460	119,000	57	965,058,046,960	603,043,436,904
17	244,358	231,844	58	831,765,594,568	509,897,148,964
18	475,468	447,342	59	684,759,208,366	412,039,723,036
19	914,426	859,744	60	538,359,358,490	317,373,604,363
20	1,755,070	1,637,383	61	401,797,523,128	232,306,415,924
21	3,327,004	3,098,270	62	285,131,794,258	161,303,043,901
22	6,278,248	5,802,411	63	190,679,390,020	105,730,020,222
23	11,702,798	10,783,780	64	120,772,653,396	65,450,375,310
24	21,684,174	19,826,318	65	71,454,350,152	37,942,606,582
25	39,674,570	36,142,146	66	39,922,745,752	20,696,691,144
26	72,079,114	65,135,623	67	20,599,350,216	10,460,286,822
27	129,231,886	116,238,056	68	10,019,242,068	4,961,671,731
28	229,705,420	204,900,019	69	4,426,822,772	2,144,789,574
29	402,601,352	357,071,928	70	1,840,952,666	868,923,831
30	698,293,484	613,926,161	71	677,567,466	311,901,840
31	1,193,224,016	1,042,022,040	72	234,051,602	104,859,366
32	2,014,400,784	1,742,855,397	73	68,291,438	29,592,634
33	3,346,429,688	2,873,077,198	74	18,421,746	7,766,947
34	5,482,511,216	4,660,800,459	75	3,758,728	1,508,596
35	8,827,837,774	7,439,530,828	76	683,042	272,198
36	13,992,014,012	11,668,443,776	77	70,550	26,638
37	21,766,319,560	17,976,412,262	78	8,168	3,406
38	33,266,838,564	27,171,347,953	79	180	70
39	49,831,763,900	40,271,406,380	80	34	17
40	73,199,529,058	58,469,060,820	Sum	15,716,295,466,894	10,461,394,944,000

17 Positions @ Depth 80

Solution for FA8CBE9D72513640



Solving Single Instances

- Shortest Path
- Admisible Heuristic
  - Estimates the Distance to the start Position
  - Will never overestimate the Distance
  - Used for pruning Nodes
- 32 Opteron Cores + 256GB main memory

Demo

Applications

- All Kinds of Optimization Problems
- DNA Sequence Alignment
- Model Checking
- …

References

- A. Reinefeld, T. Schütt. *Out-of-Core Parallel Heuristic Search with MapReduce*. High-Performance Computing Symposium HPCS 2009, Kingston, Ontario.

- Korf et al. (2000, 2005, …): frontier search, delayed duplicate detection
- Zhou et al. (2004, …): BFHS, BF-IDA*, structured duplicate detection
- Edelkamp et al. (2004): External A*
- Culberson et al. (1994, ..): pattern databases

Thanks!
Questions?