New Ideas Track: Testing MapReduce-Style Programs

Christoph Csallner, Leonidas Fegaras, Chengkai Li

Computer Science and Engineering Department
University of Texas at Arlington (UTA)

European Software Engineering Conference / ACM SIGSOFT
Symposium on the Foundations of Software Engineering

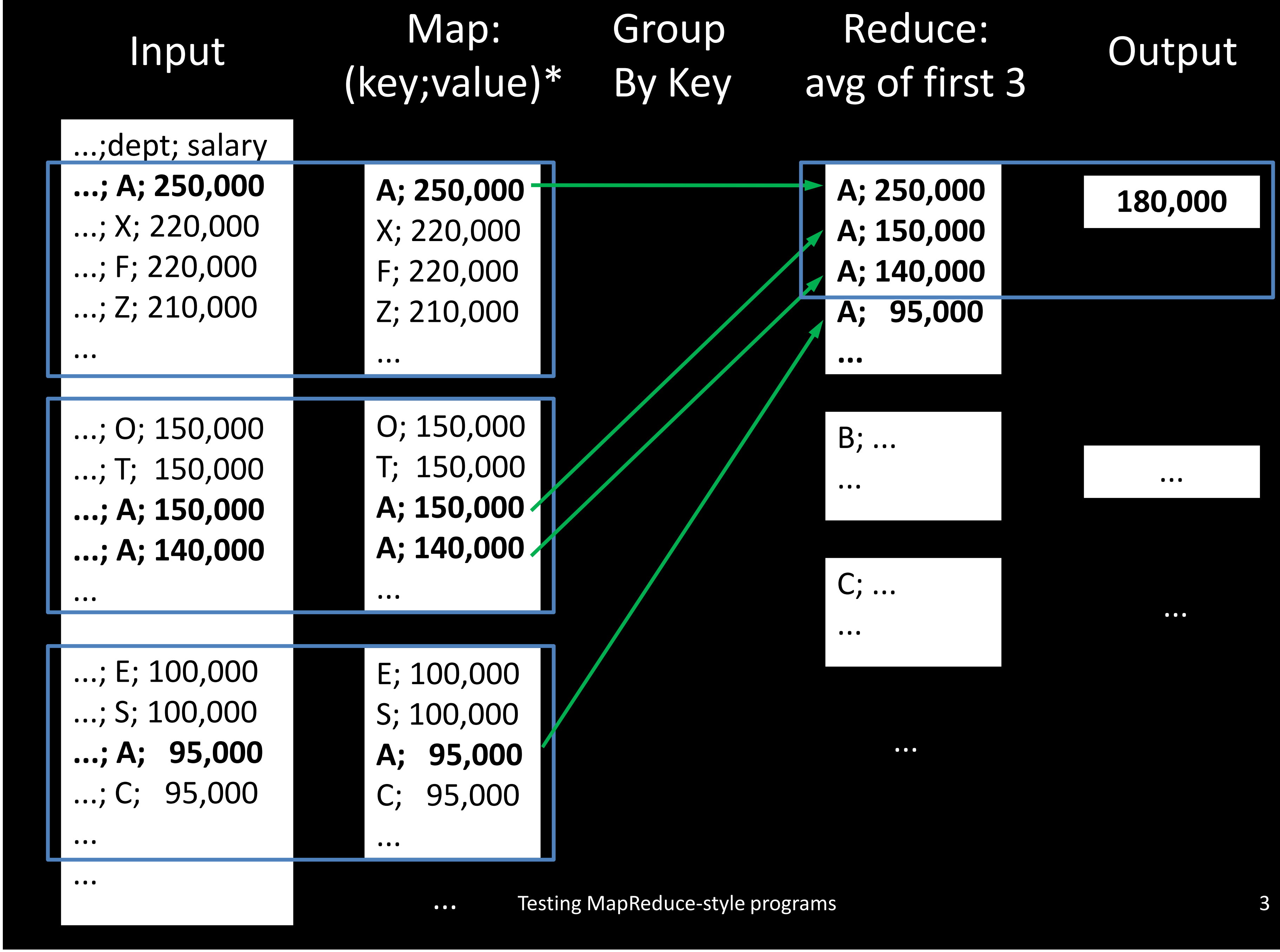
ESEC/FSE, Szeged, Hungary, Thursday, Sep. 8, 2011

Since 2004: Many MapReduce systems, papers & users

- Google MapReduce [OSDI 2004] > 2,000 cit.
- Apache/Yahoo! Hadoop
  - http://wiki.apache.org/hadoop/PoweredBy
- Microsoft Dryad [EuroSys 2007] > 500 cit.
  - http://research.microsoft.com/en-us/projects/dryad/
- Apache/Yahoo! Pig [SIGMOD 2008] > 400 cit.
  - https://cwiki.apache.org/confluence/display/PIG/PoweredBy
- Apache/Facebook Hive [VLDB 2009]
  - https://cwiki.apache.org/confluence/display/Hive/PoweredBy

MapReduce programming model

- **Programmer implements sequential code**
  - Two functions: map and reduce
  - For example, in sequential Java code
- System distributes, schedules, handles faults
  - Invokes map **on many nodes in parallel**
  - Collects and re-distributes intermediate results
  - Invokes reduce **on many nodes in parallel**
- Programmer can focus on problem domain



Input

...;dept; salary

...; A; 250,000
...; X; 220,000
...; F; 220,000
...; Z; 210,000
...

...; O; 150,000
...; T; 150,000
...; A; 150,000
...; A; 140,000
...

...; E; 100,000
...; S; 100,000
...; A; 95,000
...; C; 95,000
...

...

Map:
(key;value)*

A; 250,000
X; 220,000
F; 220,000
Z; 210,000
...

O; 150,000
T; 150,000
A; 150,000
A; 140,000
...

E; 100,000
S; 100,000
A; 95,000
C; 95,000
...

...

Group
By Key

Reduce:
avg of first 3

A; 250,000
A; 150,000
A; 140,000
A; 95,000
...

B; ...
...

C; ...
...

...

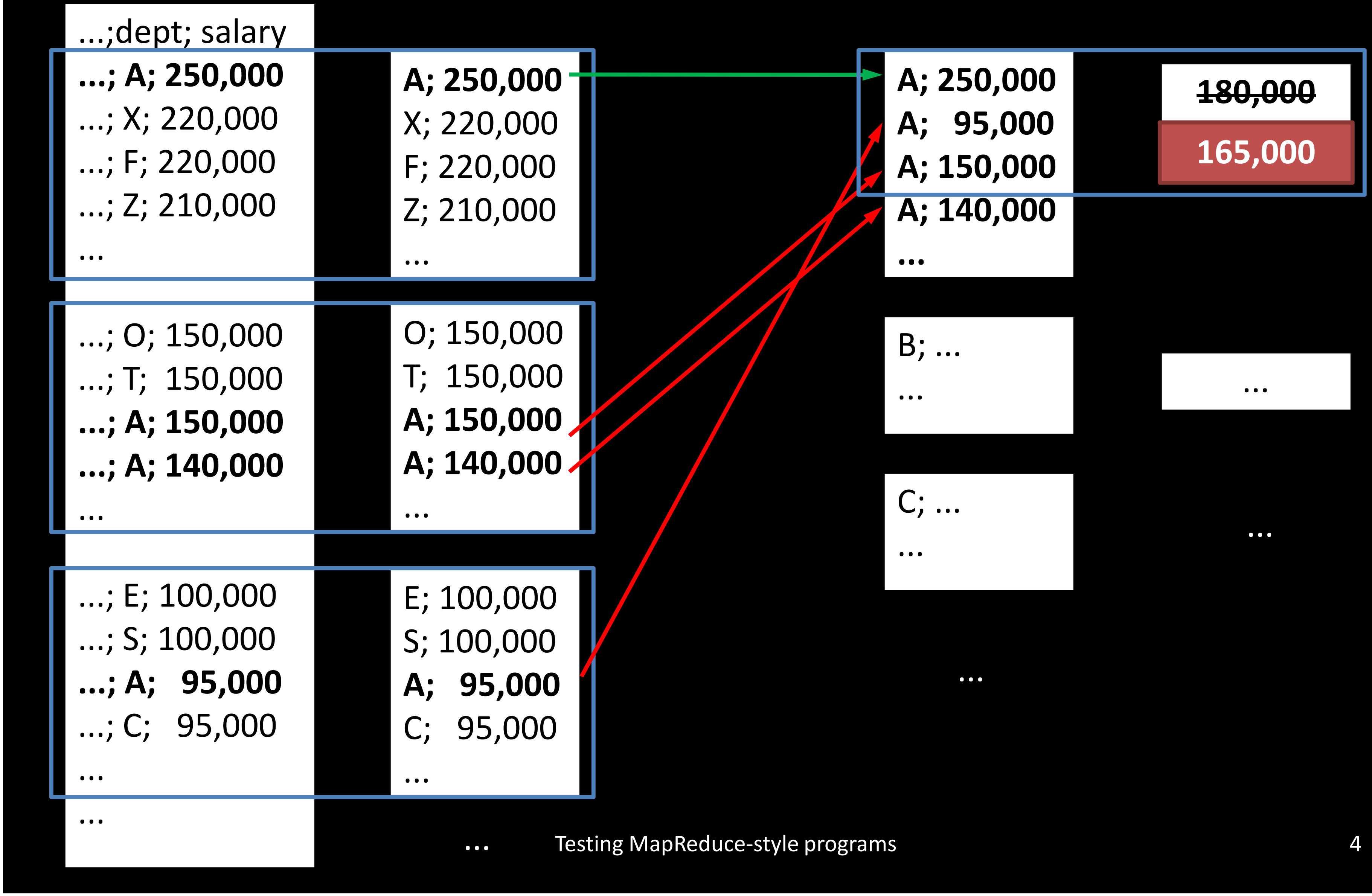
Output

180,000

...

...

Input | Map: (key;value)* | Group By Key | Reduce: avg of first 3 | Output



…;dept; salary
…; A; 250,000
…; X; 220,000
…; F; 220,000
…; Z; 210,000
…

…; O; 150,000
…; T; 150,000
…; A; 150,000
…; A; 140,000
…

…; E; 100,000
…; S; 100,000
…; A;   95,000
…; C;   95,000
…
…

A; 250,000
X; 220,000
F; 220,000
Z; 210,000
…

O; 150,000
T; 150,000
A; 150,000
A; 140,000
…

E; 100,000
S; 100,000
A;   95,000
C;   95,000
…

…

A; 250,000
A;   95,000
A; 150,000
A; 140,000
…

B; …
…

C; …
…

…

~~180,000~~
165,000

…

…

Example bug:

```
/* Report avg of top-3 salaries, if avg>100k */
public void reduce(String dept, Iterator<Integer> salaries) {
    int sum = 0; int i = 0;
    while (salaries.hasNext() && i<3) {
        sum += salaries.next();
        i += 1;
    }
    emit( (i>0 && sum/i > 100000)? sum/i : -1);
}
```

- Code depends on order of salaries, just uses first-3
- Programmer may be confused by order of salaries in input files, that order is not maintained
- Bug, possibly because MapReduce systems have built-in ordering, but not always use them

User reduce program has to satisfy correctness conditions

- Reduce must not rely on a particular order:
- For each input list of values L,
  for each permutation P:
  reduce(key, L) == reduce(key, P(L))

- Program also has to satisfy other MapReduce-specific correctness conditions
- Current tools do not check these conditions

Goal: Find such bugs automatically

- Find an input list of values L and a permutation P:
  reduce(key, L) ≠ reduce(key, P(L))
- Current tools do not find such bugs
- There are many input lists and permutations
  - Trying all of them is impossible

Example bug:

```
/* Report avg of top-3 salaries, if avg>100k */
public void reduce(String dept, Iterator<Integer> salaries) {
   int sum = 0; int i = 0;
   while (salaries.hasNext() && i<3) {
      sum += salaries.next();
      i += 1;
   }
   emit( (i>0 && sum/i > 100000)? sum/i : -1);
}
```

- Need specific list of salaries & permutation
  - List of more than 3 elements
  - Average of first 3 elements > 100k
  - Permutation has to swap element at position≤3 with element at position>3

Observations

- Example MapReduce programs are typically small and contain few execution paths
  - How do industrial MapReduce programs look like?
- Dynamic symbolic execution may be a good fit
  - Heavy-weight but precise analysis
  - Systematically explores all execution paths
  - Well-suited for reasoning about few paths
- reduce(key, L), reduce(key, P(L)) may trigger different execution paths
  - Not enough to analyze one path at a time

Check correctness conditions with dynamic symbolic execution

1. Derive symbolic path condition, return value
2. Maintain them in an indexed execution tree
   - Index leaf nodes by length of input list
   - Sibling(path): Triggered by input list of same length
3. Encode potential violation of correctness condition in constraint system
   - Solving constraints with off-the-shelf constraint solver yields concrete input values L and permutation P
4. Convert solution to test case, run, confirm violation

Encode correctness conditions in symbolic program constraints

```
// Permutation P as a function: 0 → p[0], 1 → p[1], ..
// Symbolic list L = L[0], L[1], ..        P(L) = L[p[0]], L[p[1]], ..
SymbolicInt[] p ← SymbolicIndices;   // distinct list positions
Assert PathCond;                                 // e.g.: L[0]==5
Assert SubstituteIndices(SiblingPath, p);  // e.g.: L[p[0]]==5

// Find a concrete list + a concrete permutation such that:
//   reduce(key, list) ≠ reduce(key, permutation(list))
Assert Result ≠ SubstituteIndices(SiblingResult, p);
```

Input length heuristic

- Pick “representative” input lengths
- Initially: |L| := 2
  - For shorter lists: L == P(L)
- Binary back-off scheme
  - Each subsequent iteration doubles length of L

Conclusions

- New programming paradigm with new bugs
  - To produce deterministic results, a MapReduce system requires user programs to satisfy certain high-level correctness conditions
  - Neither MapReduce execution systems nor tools check these conditions
- Proposed approach:
  - Encode MapReduce correctness conditions in symbolic program constraints
  - Check correctness conditions at runtime

References

- **[OSDI 2004]** J. Dean and S. Ghemawat. *MapReduce: Simplified data processing on large clusters*. In Proc. 6th USENIX Symposium on Operating Systems Design and Implementation, pages 137—150.
- **[EuroSys 2007]** M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. *Dryad: Distributed data-parallel programs from sequential building blocks*. In Proc. 2nd ACM SIGOPS European Conference on Computer Systems, pages 59—72.
- **[SIGMOD 2008]** C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. *Pig latin: A not-so-foreign language for data processing*. In Proc. 34th ACM SIGMOD International Conference on Management of Data, pages 1099—1110.
- **[CACM2008]** J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1):107—113.
- **[VLDB 2009]** A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wycko, and R. Murthy. *Hive: A warehousing solution over a map-reduce framework*. Proc. VLDB Endowment, 2(2):1626—1629.

Questions

MapReduce used for variety of jobs

- Process “web-scale” data (PB = peta-byte = 10^{15})
  - Run on many machines in parallel
- Google: Process 20 PB per day [CACM2008]
  - 10k programs build search index, process text, graphs, etc.
- New York Times: Convert 4TB of articles to PDF
  - http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/
- Yahoo!: Sort TB in 209 seconds: http://sortbenchmark.org/
  - “First time that either a Java or an open source program has won this challenge” [http://hadoop.apache.org/]
- Facebook: Hive-based data warehouse

MapReduce ≠ map-reduce

- MapReduce:
  - Inspired by functional programming map-reduce
  - But different ☺
- For detailed comparison, see:
  - Ralf Lämmel. *Google's MapReduce programming model — Revisited*. Science of Computer Programming 68(3): 208—237. Oct. 2007.

MapReduce correctness condition 2: Optional combine function

- Combine: programmer-defined sequential code
  - Similar to map and reduce
- May be invoked on Map node, after map
  - Locally “pre-reduce” results, by key
  - Reduce transmission overhead to “real reduce”
- System can invoke combine 0—n times
  - Must not affect semantics
- Similar approach:
  - Encode in symbolic path condition, result value