BatchSolve: A Divide and Conquer Approach to Solving the Memory Ordering Problem

Debarshi Chatterjee, Ismet Bayraktaroglu, Nikhil Sathe, Kavya Shagrithaya, Siddhanth Dhodhi, Spandan Kachhadiya
Memory Ordering Problem

• Given the Issue Order of MemOps, can we find a Global Order that satisfies all the Ordering Rules?

• Notion of Global Order: To an external observer, the MemOps appear to happen in this Order.

• Ordering Rules
  • Memory Consistency Models
  • Deadlock Avoidance Rules
  • Micro-Architectural Specifications

Example Ordering Rule: MemOps from same source must appear in GO in the same order as the appear in issue order
Prior Work

- Naïve Approaches
- Point of Serialization Snooping (POSS)
  - Basic Idea: Get hints from RTL to obtain GO
  - Pros: Good Coverage, Linear Time Complexity
  - Cons: Portability, Dev. & maintenance cost
- TSO Tool [Hangal et al., ISCA 2004]
  - Assumption: Unique store Data
  - Basic Idea: Construct Graph and check for cycles
  - Pros: Reasonably good coverage, Polynomial Time Complexity
  - Cons: Random atomics, not amenable to arbitrary ordering rules

![Execution 2: Illegal](image)

<table>
<thead>
<tr>
<th>SRC1</th>
<th>SRC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST₁(A)X</td>
<td>LD₁(A)Y</td>
</tr>
<tr>
<td>ST₂(A)Y</td>
<td>LD₂(A)X</td>
</tr>
</tbody>
</table>

**Edge Color Coding**
- Blue: Issue Order Dep. Edges
- Green: Observed Dep. Edges
- Orange: Inferred Edges
Motivation and High-Level Idea

• Need an ordering checking scheme with following:
  • Portable (Horizontal and Vertical Re-use)
  • Low Cost (Low Dev. And maintenance Effort; Immune to Arch. changes)
  • Flexible (Amenable to various ordering rules – PCIE, NVLink, other Link based rules)

• BatchSolve - High Level Idea:
  • Specify Ordering Rules as high-level SV constraints
  • Formulate the problem so that SV solver can handle it
  • How do we address scalability issues? – Stimulus Batching
BATS – Integration into UVM Architecture

Diagram showing the integration of BATS into the UVM architecture, with components labeled as follows:
- `test`
- `env`
- `scoreboard`
- `Agent1`, `Agent2`, `AgentN`
- `Interface`
- `DUT`
- `Batch-Solve`
- `BATS- Stimulus API`
- `Sequence`
- `Sequencer`
- `Monitor`
- `Driver`
BATS – Stimulus Batching

$B^X_0$  

Cacheline X transition to almost full state

$B^X_1$

Spaced-read: Cacheline X transition to full state

$B^Y_0$

Cacheline X: ready to use again

$B^Y_1$


$X, Y$: Cacheline address  
$B^X_i$: Batch $i$ to address $X$  
$B^Y_i$: Batch $i$ to address $Y$

Arrows: MemOp drive time from across sources  
$\Delta$: Time Threshold
### Stimulus Batching - Continued

<table>
<thead>
<tr>
<th>Issue-Time</th>
<th>SRC</th>
<th>MemOp</th>
<th>Sector-0</th>
<th>Sector-1</th>
<th>Sector-2</th>
<th>Sector-3</th>
<th>Count</th>
<th>STATE</th>
<th>Batch-num</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>SRC1</td>
<td>Wr1</td>
<td>Wr1-C0</td>
<td>Wr1-C1</td>
<td>Wr1-C2</td>
<td></td>
<td>1</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T2</td>
<td>SRC1</td>
<td>Rd1</td>
<td>Rd1-C0</td>
<td>Rd1-C1</td>
<td></td>
<td></td>
<td>2</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T3</td>
<td>SRC2</td>
<td>Wr2</td>
<td>Wr2-C0</td>
<td>Wr2-C1</td>
<td>Wr2-C2</td>
<td>Wr2-C3</td>
<td>3</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T4</td>
<td>SRC2</td>
<td>Wr3</td>
<td>Wr3-C0</td>
<td>Wr3-C1</td>
<td>Wr3-C2</td>
<td></td>
<td>4</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T5</td>
<td>SRC2</td>
<td>Rd2</td>
<td>Rd2-C0</td>
<td>Rd2-C1</td>
<td>Rd2-C2</td>
<td>Rd2-C3</td>
<td>5</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T6</td>
<td>SRC1</td>
<td>Rd3</td>
<td>Rd3-C0</td>
<td>Rd3-C1</td>
<td>Rd3-C2</td>
<td>Rd3-C3</td>
<td>6</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T7</td>
<td>SRC1</td>
<td>Wr4</td>
<td>Wr4-C0</td>
<td>Wr4-C1</td>
<td>Wr4-C2</td>
<td></td>
<td>7</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T8</td>
<td>SRC2</td>
<td>Wr5</td>
<td>Wr5-C0</td>
<td>Wr5-C1</td>
<td>Wr5-C2</td>
<td>Wr5-C3</td>
<td>8</td>
<td>EMPTY</td>
<td>0</td>
</tr>
<tr>
<td>T9</td>
<td>SRC2</td>
<td>Rd4</td>
<td>Rd4-C0</td>
<td>Rd4-C1</td>
<td>Rd4-C2</td>
<td></td>
<td>9</td>
<td>A-FULL</td>
<td>0</td>
</tr>
<tr>
<td>T10+Δ</td>
<td>SRC1</td>
<td>Rd5</td>
<td>Rd5-C0</td>
<td>Rd5-C1</td>
<td>Rd5-C2</td>
<td>Rd5-C3</td>
<td>10</td>
<td>FULL</td>
<td>0</td>
</tr>
<tr>
<td>T11+Δ</td>
<td>SRC1</td>
<td>Wr6</td>
<td>Wr6-C0</td>
<td>Wr6-C1</td>
<td>Wr6-C2</td>
<td></td>
<td>1</td>
<td>EMPTY</td>
<td>1</td>
</tr>
<tr>
<td>T12</td>
<td>SRC2</td>
<td>Wr7</td>
<td>Wr7-C0</td>
<td>Wr7-C1</td>
<td></td>
<td></td>
<td>2</td>
<td>EMPTY</td>
<td>1</td>
</tr>
<tr>
<td>SRC</td>
<td>UID</td>
<td>MemOp</td>
<td>Sector-0</td>
<td>Read Data Rcvd</td>
<td>Byte Enable</td>
<td>Write Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-------------</td>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Byte-0</td>
<td>Byte-1</td>
<td>Byte0</td>
<td>Byte1</td>
<td>Byte0</td>
<td>Byte1</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>Dummy Init Wr</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>I1</td>
<td>I2</td>
</tr>
<tr>
<td>T1</td>
<td>SRC1</td>
<td>1</td>
<td>Wr1-C0</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>X1</td>
<td>X2</td>
</tr>
<tr>
<td>T2</td>
<td>SRC1</td>
<td>2</td>
<td>Rd1-C0</td>
<td>X3</td>
<td>X4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>SRC2</td>
<td>3</td>
<td>Wr2-C0</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>X3</td>
<td>X4</td>
</tr>
<tr>
<td>T5</td>
<td>SRC2</td>
<td>4</td>
<td>Rd2-C0</td>
<td>X3</td>
<td>X4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>SRC1</td>
<td>5</td>
<td>Rd3-C0</td>
<td>X7</td>
<td>X4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>SRC1</td>
<td>6</td>
<td>Wr4-C0</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>X5</td>
<td>X6</td>
</tr>
<tr>
<td>T8</td>
<td>SRC2</td>
<td>7</td>
<td>Wr5-C0</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>X7</td>
<td>X8</td>
</tr>
<tr>
<td>T9</td>
<td>SRC2</td>
<td>8</td>
<td>Rd4-C0</td>
<td>X5</td>
<td>X6</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>SRC1</td>
<td>9</td>
<td>Rd5-C0</td>
<td>X5</td>
<td>X6</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MemOp at issued at T4 on SRC2 is omitted because it does not have a child to sector0 for which ordering is being tested.
## SV Solver – Sample Output

<table>
<thead>
<tr>
<th>Issue-Time</th>
<th>SRC</th>
<th>UID</th>
<th>GO</th>
<th>Sector-0</th>
<th>Read Data Rcvd</th>
<th>Byte Enable</th>
<th>Write Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Byte-0</td>
<td>Byte-1</td>
<td>Byte0</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T1</td>
<td>SRC1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>SRC2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>SRC1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>SRC2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T8</td>
<td>SRC2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T6</td>
<td>SRC1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T7</td>
<td>SRC1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td>SRC2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T10</td>
<td>SRC1</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Notice in UID column MemOps from same SRC appears in issue order.

Rd1 inferred to be ordered after Wr2.

Rd4 inferred to be ordered after Wr2 and Wr5.
Inside the Solver – 1

Input Matrix I:
MemOps in issue order

Row Permuted
Random Matrix P:
MemOps in GO

Ordering Rule as Constraints
Inside the Solver - 2

**Input Matrix I:** MemOps in issue order

**Row Permuted Random Matrix P:** MemOps in GO

**Random Matrix M:** Mimics memory value after exec of corresponding row of P

**Ordering Rule as Constraints**

**GO Constraints**

```
constraint c_read {
    foreach (M[i,j]){
        // Equate data from reads in GO into memory model, if corresponding byte enables are set.
        (P_be[i][j] == 1) && (P_cmd_type[i] == READ) -> (M[i][j] == P_data[i][j]);
    }
}
```
Inside the Solver - 3

Input Matrix I:
MemOps in issue order

Row Permuted
Random Matrix P:
MemOps in GO

Ordering Rule as Constraints

Random Matrix M:
Mimics memory value after exec of corresponding row of P

GO Constraints

```c
constraint c_write {
    foreach (M[i,j]){
        //Equate data from writes in GO into memory model, if corresponding byte enables are set.
        (P_be[i][j] == 1) && (P_cmd_type[i] == WRITE) -> (M[i][j] == P_data[i][j]);
    }
};
```
Inside the Solver - 4

Ordering Rule as Constraints
Input Matrix I: MemOps in issue order

Row Permuted
Random Matrix P:
MemOps in GO

GO Constraints
Random Matrix M:
Mimics memory value after exec of corresponding row of P

P_be[i][j]  P_data[i][j]  M[i][j]

```
constraint c_invariance{
    foreach (M[i,j]){
        if(i>0 && (P_cmd_type[i] == READ || (P_cmd_type[i] inside {WRITE, ATOMIC_AND} && P_be[i][j] == 0))){
            M[i][j] = M[i-1][j];
        }
    }
}
```
BatchSolve – Advanced Topics

• Atomic Handling
  • Replace Function calls with explicit SV constraints

• Barrier Handling
  • Using “Rules” determine which MemOps should be ordered before the Barrier and which should be ordered after it (some MemOps can be neither)
  • Remove the Barrier and draw edges from every MemOp in $S_{before}$ to every MemOp in $S_{after}$
  • These edges are created in pre_randomize and introduced as constraints to the SV solver

Sample Rule: *Reads issued after membar-ack from same or different source (as that of membar) must be ordered after the membar*
EDA Playground Demo Links

• **Link to simple DEMO (Reads/Writes/Atomics)**
  • [https://www.edaplayground.com/x/rXKN](https://www.edaplayground.com/x/rXKN)
  • *Example of an execution for which GO exists and one for which it does not*

• **Link to advanced DEMO (Barrier Handling)**
  • [https://www.edaplayground.com/x/DsUc](https://www.edaplayground.com/x/DsUc)
  • *Examples of legal and illegal execution with barrier*
## Results

<table>
<thead>
<tr>
<th></th>
<th>POSS</th>
<th>BATS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Init Develop effort</strong></td>
<td>80 weeks</td>
<td>8-weeks</td>
</tr>
<tr>
<td><strong>Maintenance effort per project (estimate)</strong></td>
<td>80 weeks</td>
<td>0-1 weeks (not including debug)</td>
</tr>
<tr>
<td><strong>Porting effort to other UVM TB</strong></td>
<td>Not portable easily</td>
<td>1 week (assuming TB has some score boarding)</td>
</tr>
</tbody>
</table>

![Comparison of Runtime and SimTime for POS vs BatchSolve](chart.png)
Conclusion and Future Work

• BATS Pros
  • Low Development Cost
  • Low Maintenance Cost
  • Easily Portable
  • Easy to specify ordering rules as high level SV constraints

• BATS Cons
  • Slight Coverage loss due to Batching
  • Slight increase in runtime

• Future Work
  • Can a re-formulation such as convex relaxation allow to increase Batch-Size?
Thank you!