Sloth: Being Lazy is a Virtue
When Issuing Database Queries

Alvin Cheung
Sam Madden   Armando Solar-Lezama

MIT CSAIL
Sample DB application code

```java
handleRequest (name) {
    p = getPatient(name);

    t = getBodyTemp(p);
    tc = (t - 32) * 5 / 9;

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
```

Issues query to DB
Sample DB application code

```javascript
handleRequest (name) {
    p = getPatient(name);
    t = getBodyTemp(p);
    tc = (t - 32) * 5 / 9;
    visits = getVisits(p);
    meds = getMeds(p);
    printHTML(tc, visits, meds);
}
```

Issues query to DB
handleRequest (name) {
  p = getPatient(name);
  t = getBodyTemp(p);
  visits = getVisits(p);
  meds = getMeds(p);
  tc = (t - 32) * 5 / 9;
  printHTML(tc, visits, meds);
}

Ibrahim et al, ECOOP 06
Ramachandra et al, SIGMOD 12
handleRequest (name) {  
p = getPatient(name);
if (canViewVitals) {
    t = getBodyTemp(p);
    tc = (t - 32) * 5 / 9;
    // test harness
    List getVisits (Patient p) {
        return executeQuery(p.id);
    }
}
"Do it only when you need it"

Extended Lazy Evaluation
handleRequest (name) {

    p = getPatient(name);

    if (canViewVitals) {
        t = getBodyTemp(p);
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {
    p = thunk1

    if (canViewVitals) {
        t =getBodyTemp(p);  
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
handleRequest (name) {

    p = thunk1;

    if (canViewVitals) {
        t = getBodyTemp();
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t =getBodyTemp();
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {
    p = thunk1
    if (canViewVitals) {
        t = thunk2
        tc = (t - 32) * 5 / 9;
    }
    visits = getVisits(p);
    meds = getMeds(p);
    printHTML(tc, visits, meds);
}
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Batch 1
Query: getPatient(name)
Result: 

Batch 2
Query: getBodyTemp()
Result: not yet executed

Issues query to DB
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = getVisits(thunk1);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = getVisits();
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = thunk4
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest \((\text{name})\) {

\[p = \text{thunk 1}\]

if \((\text{canViewVitals})\) {

\[t = \text{thunk 2}\]
\[\text{tc} = \text{thunk 3}\]

} 

\[\text{visits} = \text{thunk 4}\]
\[\text{meds} = \text{thunk 5}\]

printHTML(\text{tc}, \text{visits}, \text{meds});
}
handleRequest (name) {
    p = thunk1
    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }
    visits = thunk4
    meds = thunk5
    printHTML(t, tc, visits, meds);
}
handleRequest (name) {
    p = thunk1
    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }
    visits = thunk4
    meds = thunk5
    printHTML(t, tc, meds);
}
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = thunk4
    meds = thunk5

    printHTML(thunk3, thunk4, thunk5)
}
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t = thunk2
        tc = thunk3
    }

    visits = thunk4
    meds = thunk5

    printHTML(thunk3, thunk4, thunk5)
}

Query Store

Batch 1
Query: getPatient(name)
Result: 

Batch 2
Query: getBodyTemp()
Result: not yet executed

Query: getVisits()
Result: not yet executed

Query: getMeds()
Result: not yet executed
handleRequest (name) {

    p = thunk1

    if (canViewVitals) {
        t =
        tc =
    }

    visits =
    meds =

    printHTML(
    );
}

Query:
getPatient(name)
Result:

Query:
getBodyTemp()
Result:

Query:
getVisits()
Result:

Query:
getMeds()
Result:
handleRequest (name) {
    p = thunk1

    if (canViewVitals) {
        t =
        tc =
    }

    visits =
    meds =

    printHTML(
    );
}
handleRequest (name) {

  p = thunk1

  if (canViewVitals) {
    t =
    tc =
  }

  visits =
  meds =

  printHTML(
    ,
    ,
  );
}
handleRequest (name) {
    p = thunk1
    if (canViewVitals) {
        t = 
        tc = (\( t - 32 \)) * 5 / 9;
    }
    visits =
    meds =
    printHTML(tc, , , );
}
Sloth Toolchain

Compilation

Application source code & config files

Sloth Compiler

Compiled source using lazy evaluation

Execution

Query store

Sloth Runtime Libraries

Application Server

Java

XML

Java

Java

Compilation

Execution
class Thunk<T>
{
    T force()
    {
        if (!evaluated) {
            r = <original program expression>
            evaluated = true;
        }
        return r;
    }
}
handleRequest (name) {
    p = getPatient(name);
    if (canViewVitals) {
        t = getBodyTemp(p);
        tc = (t - 32) * 5 / 9;
    }
    visits = getVisits(p);
    meds = getMeds(p);
    printHTML(tc, visits, meds);
}

Issues query to DB
handleRequest (name) {

    p = getPatient(name);

    if (canViewVitals) {
        t = getBodyTemp(p);
        \[ tc = (t - 32) \times 5 / 9; \]
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
handleRequest (name) {

    p = getPatient(name);

    if (canViewVitals) {
        t = getBodyTemp(p);
        tc = (t - 32) * 5 / 9;
    }

    visits = getVisits(p);
    meds = getMeds(p);

    printHTML(tc, visits, meds);
}
When to force

Field writes
\[ x.f = y \quad \rightarrow \quad x.\text{force}().f = y; \]

Control flow
\[ \text{if} \ (c) \ \{ \ … \ \} \quad \rightarrow \quad \text{if} \ (c.\text{force}()) \ \{ \ … \ \} \]

Statements with externally-visible side effects
\[ \text{print}(x); \quad \rightarrow \quad \text{print}(x.\text{force}()); \]

Data modification / transaction statements
\[ \text{execute}(\text{“INSERT INTO …”}) \]
\[ \rightarrow \quad \text{queryStore.executeBatchedQueries}(); \]
\[ \quad \text{queryStore.execute}(\text{“INSERT INTO …”}); \]
Optimizations

Selective lazification

Coalesce thunks

Defer control flow blocks and method calls
Defer control flow blocks

```java
if (c)
    x = a;
else
    x = b;
```

```
if (c.force())
    x = new Thunk(a) { force() { return a.force(); } };
else
    x = new Thunk(b) { force() { return b.force(); } };
```

```
x = new Thunk(c, a, b) {
    force() {
        if (c.force())
            return a.force();
        else
            return b.force();
    }
};
```
Experiments

Two large-scale open source web apps
Extracted 38 and 112 benchmarks

Hosted on tomcat and mysql
0.5ms round trip ping time

Goals
Page load times
Scaling effects
Runtime overhead
Sloth speeds up most benchmarks

Ratio = Original page load time / Sloth page load time

Max: 2.1x
Median: 1.15x
Sloth performs even better when network round trip time increases.

3.1x speedup @ 10ms
Batching queries increases throughput

Max: 1.5x

CPU bound @ web server

I/O bound @ DB

Throughput (page/s)

Number of clients
# Runtime overhead

<table>
<thead>
<tr>
<th>Txn type</th>
<th>Original (s)</th>
<th>Sloth (s)</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPC-C</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New order</td>
<td>930</td>
<td>955</td>
<td>15.8%</td>
</tr>
<tr>
<td>Order status</td>
<td>752</td>
<td>836</td>
<td>11.2%</td>
</tr>
<tr>
<td>Stock level</td>
<td>420</td>
<td>459</td>
<td>9.4%</td>
</tr>
<tr>
<td>Payment</td>
<td>789</td>
<td>869</td>
<td>10.2%</td>
</tr>
<tr>
<td>Delivery</td>
<td>626</td>
<td>665</td>
<td>6.2%</td>
</tr>
<tr>
<td><strong>TPC-W</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browsing mix</td>
<td>1075</td>
<td>1138</td>
<td>5.9%</td>
</tr>
<tr>
<td>Shopping mix</td>
<td>1223</td>
<td>1326</td>
<td>8.5%</td>
</tr>
<tr>
<td>Ordering mix</td>
<td>1423</td>
<td>1600</td>
<td>12.4%</td>
</tr>
</tbody>
</table>

6 – 16 %
Extended lazy evaluation
Batch queries issued by client
3x performance speedup

Alvin Cheung  akcheung@mit.edu