ISTC-CC Presentation by Georgia Tech (8/27/15)

Karsten Schwan (site director), Calton Pu (presenter), with L. Liu, S. Yalamanchili, G. Eisenhauer, A. Gavrilovska, M. Wolf

Georgia Tech

http://www.istc-cc.cmu.edu/
Highlights of ISTC-CC (Georgia Tech)

• First part: yearly update on various projects
  ▫ Karsten Schwan, Ling Liu, Calton Pu, Sudha Yalamanchili, Greg Eisenhauer, Ada Gavrilovska, Matt Wolf
  ▫ Many students at PhD, MS, and undergraduate levels

• Second part: some details on automated management work (linking into Project Pulse)

• Highlights on work funded by ISTC-CC, or conforming to Intel open IP policy
  ▫ Significant funding amplification from many industry collaborators and government funding agencies such as NSF and DoE
GraphReduce Architecture (Schwan)

SC 2015, GraphReduce: Processing Large-Scale Graphs on Accelerator-Based Systems
Optimizations on GraphReduce

- Asynchronous execution and Spray (deep-copy) operation
- Dynamic frontier management
- Dynamic phase fusion and elimination
Gains by GraphReduce

GraphReduce’s speedup over Graphchi and X-Stream for out-of-memory graph inputs

Benefits of GraphReduce optimizations over memcpy time
Understanding Issue Correlations: A Case Study of the Hadoop System

Scalable distributed systems are complex [Yuan et al., OSDI’14]

Complicated System + Error-prone + Hard to Debug → Issue Study

Issue Pattern → Better Software & Debugging Tools

SOCC’15
Our Findings

- Half of the issues are independent
- The issue correlations are not complex as we expected
- One third of the issues have similar causes
- ......

- For memory issues, GC is still the No. 1 concern
- The statement “99.99% of data reliability” is challenged
- One third of programming issues relate to interfaces
- The logging system is error-prone
- ......
Methodology Used in Our Study

- Computation
- Storage

- Hive
- HCatalog
- Pig
- Mahout
- Flume
- Cascading
- Mahout
- MapReduce
- Hive
- Pig
- Flume
- Cascading

- Description
- Patches
- Follow-up Discussions
- Source Code Analysis

- IssueID
- Subcomponent
- Type
- Causes
- CorrelatedIssueID

- Create/Commit Time
- HPatchDB

- Labeling

Sampling Period: ~6 years
Sampling Rate: 89.8%

- IssueID
- Create/Commit Time
- Subcomponent
- Type
- Causes
- CorrelatedIssueID
- ……

Closed Issue
Examined Issue

Issue Description
Patches
Follow-up Discussions
Source Code Analysis
1. **Correlations Between Issues**
   Issues are independent; 33% of issues have similar causes, etc.

2. **Correlations With System Characteristics**
   Systems, programming, tools
• Optimizations for Fast Iterative Graph Computations
  ▫ GraphLego:
    • Resource Aware Graph Parallel Abstractions (Graph Cube and Slice, Strip, Dice) [ACM HPDC 2015]
  ▫ GraphTwist:
    • Approximation with utility-aware pruning [VLDB2015]
    • Edge pruning by slices: removing some insignificant edges
    • Vertex pruning by cuts: removing some insignificant vertices
  ▫ GraphMap:
    • Workload aware Distributed Graph Processing Framework [IEEE SC2015]
Shared Memory Optimization in Virt. Cloud

- **Shared Memory Management Mechanisms**
  - **MemPipe:**
    - Shared memory channels for improving communication efficiency between co-resident VMs
    - Incremental shared memory management
  - **MemFlex**
    - Shared memory based ballooning (inflate and deflate)
    - Shared memory based optimization for memory page fault.
  - **MemMon**
    - Memory working-set monitoring and estimate.

Poster by Qi Zhang (L. Liu)
Optimizing Performance and Productivity on Heterogeneous Processors

Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology

Collaborators: H. Wu, M. Gupta, C. Kersey, H. Kim, I. Saeed, J. Young, H. Wu, and LogicBlox Inc.

http://www.istc-cc.cmu.edu/
Accelerating Relational Processing

Finding cliques

- triangle\((x,y,z)\) \(\leq E(x,y), E(y,z), E(x,z)\), \(x<y<z\).
- 4cl\((x,y,z,w)\) \(\leq E(x,y), E(x,z), E(x,w), E(y,z), E(y,w), E(z,w)\), \(x<y<z<w\).

Multi-predicate Join

- Relational computations over out of core data sets
- Implementation of multi-predicate join for graph processing using GPUs
  - 3-clique and 4 clique problems

Out-of-Core Data Management

Data partitions (boxes) to fit in memory (worst-case optimal)

Box \[[3 \leq x \leq 5, 6 \leq y < \infty, -\infty < z < \infty]\]

- Large, out of core graphs
- Baseline is CPU boxed multi-predicate join
- SSD and PCIe are not the bottlenecks
Move Analytics Primitives (RA) into the memory system
- Custom low power GPU (Harmonica)
- Progress on Base compiler for in-memory GPU

Technology Assessment
- Collaboration with Lexis Nexis
- Assess the impact of In-memory acceleration for HPCC
Leveraging eBoxes and Compilers

Ada Gavrilovska
Georgia Tech

http://www.istc-cc.cmu.edu/
Enhancing the Edge via eBox-based Services

- Some results presented last year, also supported by ISTC-EC in the past and VMware
- Leverage high-density/low-power edge boxes – eBoxes
- Infrastructure for app streaming, caching, ephemeral app delivery;
- Fully integrated in Android stack
- Outcome: 2x faster app delivery, 10x faster app discovery, 70% reduction in traffic; no performance impact
- AppFlux: Taming App Delivery @TRIOS’15 (Bhardwaj, Agarwal, Gavrilovska, Schwan); others in submission/preparation
Compiler-Assisted Resource Management

- **Goal:** dynamic resource allocation to concurrent workloads/workload components
- **Problem:** profile-based techniques limited effectiveness (input-dependent requirements, irregular applications...)
- **Approach:** LLVM-based compiler infrastructure to instrument binary with “beacons”. Beacons generate information based on dynamic input and actual execution path taken. Intercepted by resource managers (e.g., VM manager, VCPU or thread scheduler, runtime-level scheduler...)
- **Outcome:** improved workload performance, reduced performance variability, improved resource use and management efficiency
- **Compiler-assisted Load Balancing on Large Clusters @PACT’15 (Deodhar, Parikh, Gavrilovska, Pande); others in submission/preparation
Automated Cloud Management through Experimental Measurements

Calton Pu
Professor and J.P. Imlay Chair in Software
Georgia Institute of Technology
Many PhD, MS, Undergraduate students and industry collaborators

http://www.istc-cc.cmu.edu/
Elba: Automated Measurements

(0) Config. Design
- Benchmark specs
- Experiment Spec. Lang.

(1) Code Generation / Deployment

(2) Execution
- Automated, Staging Cycle
- Analyzer

(3) Analyzer
- Workload Drivers
- System Under Test

(4) Reconfiguration
- Analyzed Result
- Automated Adaptation
- Adapt. Cost

Evaluation / Analysis
例实验：RUBBoS 基于 Slashdot 的基准测试

- 示例配置（1/2/1/2）
Experimental studies analyzing performance data
- Production-scale experiments on “real data centers”
- Collaboration with many industry partners
- Funding amplification from NSF

Between 2013 and 2014: 13 papers
- IEEE CLOUD, SCC, ICDCS, IRI, Big Data Congress, BigData, ACM TRIOS
Automating Experiments

- Transform and generate scripts to automatically create, manage and analyze experiments from user-friendly specification files

- Develop open tools for automated experiments
  - Support a wide variety of *evolving* clouds, benchmarks and performance monitors
  - Support flexible customization for many configuration parameters
  - High resolution monitoring at low cost
Example: Very Short Bottlenecks (TRIOS’13)

- High resolution monitoring at low cost:
  - See VSBs at tens of milliseconds
  - A few percent monitoring overhead

P-I-T Response time at 50ms resolution

Cumulative request response time distribution

Log scale

Long requests > 2%

80ms
Five Steps of Experimental Process

1. **Input Experiment Metadata (XML)**
2. **Generate Experiment Scripts**
   - User-provided configuration
   - Scripts and generator execution
3. **Execute experiment on various clouds**
4. **Collect, extract, load experimental Data**
   - Provision environment
   - Run benchmark
   - Setup, tear down infrastructure
   - Performance data extraction & load into database
5. **Analyze Results (Excel and statistical tools)**
## Scale of Experiments

*Figures for Fall 2014 and Spring 2015; taking into account diversity of work, including large-scale experimentation and infrastructure development activities*

<table>
<thead>
<tr>
<th></th>
<th>Emulab</th>
<th>PProbE</th>
<th>Local Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experiments (mins)</strong></td>
<td>91,728</td>
<td>4,641</td>
<td>2,925</td>
</tr>
<tr>
<td><strong>Nodes Used (#)</strong></td>
<td>6,048</td>
<td>1,092</td>
<td>4,516</td>
</tr>
<tr>
<td><strong>Experiments (#)</strong></td>
<td>14,112</td>
<td>714</td>
<td>450</td>
</tr>
</tbody>
</table>

*Experimental workloads range from 3 - 7 mins, each lasting about 20 - 30 min*
Step 2: Script Transformation Example

Code Template
source set_elba_env.sh
mkdir -p <xsl:value-of
select="//params[@name='RUBBOS_TOP']/@value"/>

XML Input
<xtbl name="Rubbos" version="0.1">
  <params name="OUTPUT_HOME" value="/opt/rubbos/output"/>
  <params name="RUBBOS_TOP" value="/mnt/rubbos"/>
</xtbl>

1 template line

Intermediate Representations

Occurs in 5 templates

Experiment-specific Scripts
#/home/scripts/TOMCAT_DEPLOY.sh
  source set_elba_env.sh
  mkdir -p /mnt/rubbos

And 3 script files
The following figures correspond to deploying a 16-node, (4 clients; 2W\4A\1M\4D), RUBBoS application benchmark in the Emulab cluster. Generated lines are an intermediate representation that enable application, DBMS, OS and user-specific customizations to be applied.

<table>
<thead>
<tr>
<th></th>
<th>Templates (XSLT Lines)</th>
<th>Intermediate (XML/XLST Lines)</th>
<th>Final Scripts (Shell Script Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>900</td>
<td>400</td>
<td>1500</td>
</tr>
<tr>
<td>Deployment</td>
<td>3300</td>
<td>2000</td>
<td>2200</td>
</tr>
<tr>
<td>Benchmark</td>
<td>1400</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>
Step 4: Extract/Load Monitor Results

- **Challenge:** many performance monitors, many configuration parameters, many output formats
- **Automated Approach:**
  - use script transformation techniques to annotate monitor output
  - generalize parser to consume schema (from the annotations) and parse the encapsulated data accordingly
Some monitors can output simple, CSV-formatted data files

Example 1: dstat
Example 2: sar

- Other monitors can produce highly variable and difficult-to-parse output (syntax & semantics)

```
<table>
<thead>
<tr>
<th>Time</th>
<th>CPU</th>
<th>%user</th>
<th>%nice</th>
<th>%system</th>
<th>%iowait</th>
<th>%steal</th>
<th>%idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:11:18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>all</td>
<td>1.01</td>
<td>0.00</td>
<td>0.25</td>
<td>0.50</td>
<td>0.00</td>
<td>99.24</td>
</tr>
<tr>
<td>08:11:19</td>
<td>proc/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>cswhc/s</td>
<td>2.04</td>
<td>2088.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>pswpin/s</td>
<td>pswpout/s</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>ppmg/s</td>
<td>ppmgout/s</td>
<td>fault/s</td>
<td>majflt/s</td>
<td>pgfree/s</td>
<td>pgscank/s</td>
<td>pgscand/s</td>
</tr>
<tr>
<td>08:11:19</td>
<td>tps</td>
<td>rtps</td>
<td>wtaps</td>
<td>bread/s</td>
<td>bwrt/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>frmpg/s</td>
<td>bufpg/s</td>
<td>campg/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08:11:19</td>
<td>kbenfree</td>
<td>kbenused</td>
<td>%memused</td>
<td>kbuffers</td>
<td>kcachded</td>
<td>kbccommit</td>
<td>%commit</td>
</tr>
</tbody>
</table>
```

Example 2:
sar
Parsing the following version of SAR output is reduced to parsing a XML tree

SAR Annotated Output

Transforming Output of sar

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sysstat PUBLIC "DTD v2.13 sysstat //EN" "http://pagesperso-orange.fr/sebastien.godard/sysstat.dtd">
<sysstat>
  <sysdata-version>2.13</sysdata-version>
  <host nodename="node-5.111-ba.eba.marmot.pdl.cmu.local"/>
    <osname>Linux</osname>
    <release>2.6.43.8-1.fc15.x86_64</release>
    <machine>x86_64</machine>
    <number-of-cpus>2</number-of-cpus>
    <file-date>2015-07-19</file-date>
    <statistics>
      <timestamp date="2015-07-19" time="01:16:04" utc="1" interval="1">
        <cpu-load>
          <cpu number="all" user="6.63" nice="9.00" system="3.06" iowait="0.00" steal="0.00" idle="90.31"/>
          <cpu number="0" user="13.40" nice="0.00" system="5.15" iowait="0.00" steal="0.00" idle="81.44"/>
          <cpu number="1" user="0.00" nice="0.00" system="1.61" iowait="0.00" steal="0.00" idle="98.99"/>
        </cpu-load>
        <process-and-context-switch per="second" proc="2.00" cswhc="1598.00"/>
        <swap-pages per="second" pswpin="0.00" pswpout="0.00"/>
      </timestamp>
    </statistics>
  </host>
</sysstat>
```
Research on big data graph algorithm optimization
  ▫ K. Schwan, L. Liu

Research on program optimization for heterogeneous processors and memories
  ▫ S. Yalamanchili, A. Gavrilovska

Research on automating experiments on large scale benchmarks
  ▫ C. Pu

Many publications, some tool releases, more planned