BOSS: Building Operating System Services

Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David Culler

Computer Science Division
University of California, Berkeley
© 2013 All rights reserved
Sutardja-Dai Hall
UC Berkeley
93,000 sq. ft.
with Digital Controls

73% of US electricity is consumed in buildings
U.S. Energy Information Administration, 2009

2/3 of building occupants are uncomfortable
UC Berkeley CBE Study of 30,000 occupants

>70% of large buildings have digital controls
12 Variable Speed Fans

138 Air Dampers

312 Light Relays

50 Electrical Sub-meters

151 Temperature Sensors

> 6,000 Sense and Control Points
Applications

Ventilation Optimization:
17% energy savings

Automated Fault Detection:
10 - 40% energy savings

Occupant Lighting Controls
50-60% savings

Fig. 4. Emulation study AHU recirculation damper stuck closed.
Goals and Challenges

• Portability
  – Write once, run anywhere for buildings?
  – Current practice: hand-coded logic

• Fault tolerance
  – Partial failures of controllers
  – Network partitions
  – Current practice: really tough hardware

• Multiple processes
  – Concurrent applications and users
  – Current practice: none

• Federation
  – Multiple heterogeneous systems
  – Current practice: lots of stovepipes

• Scale

• Security & privacy
BOSS: Building Operating System Services

Security
Fault tolerance
Isolation + Scheduling
History
Abstraction

Hardware Abstraction Layer

Auth.
Time-series
Trans. mgr.

HPL
HPL
HPL
HPL

Control processes

“Kernel” interface

4/5/13
NSDI 2013: Lombard, IL
Challenge: Portability

*Buildings are custom designed*
Hardware Abstraction
Physical view

Open area 450
Hardware Abstraction
Controls view

Air

\[\text{Controller}\]

\[\text{Damper} \rightarrow \text{Reheat coil}\]

SDH.MEC-08.S4-21:DMPR COMD
device: 220018 instance: 101

SDH.MEC-08.S4-21:VLV COMD
device: 220018 instance: 102

legacy solution: overload point names
Hardware Abstraction Layer

#VAV > $(120, 20)$
Summary: Hardware Abstraction Layer

Program applications in terms of *relationships between system components*

- Computer systems tend to hide the physicality
  - memory hierarchies, network topology
- Unavoidable in buildings
  - “it gets too hot on the sunny side”

Allow for scale by avoiding hard-coding

- “Run this in every room, except those on the north side”
BOSS: Building Operating System Services

- Security
- Fault tolerance
- Isolation + Scheduling
- History
- Abstraction

 Hardware Abstraction Layer

- Auth.
- Time-series
- Trans. mgr.

“Kernel” interface
Control processes

HPL

4/5/13  NSDI 2013: Lombard, IL
BOSS solution: “transactions”: write access to the building

- Writes to distributed resources
- Which interact in physical space
- Which are subject to failure
- Extend writes with
  - Priorities
  - Leases
  - Notifications
  - Reversion sequences

---

Figure 9: Ventilation component of the HV AC optimization

A completely different ventilation layout would be able to run with the HVAC system. Furthermore, a separate building with common intake setting is used to control the room ventilation requirements. Additionally, a separate building component: on-line line 6, the actual fresh air intake setting is used to control the room ventilation requirements. We demonstrate coordinated control across traditionally independent building components: on-line

The HV AC optimization control process consists of:

- Getting the current occupancy
- Calculating the required ventilation rate
- Adjusting the ventilation rates for their downstream rooms

Building codes often require a rate of fresh air ventilation per room based on occupancy and room size [10, 5]. Keeping ventilation rates at the required minimum is highly desirable for energy savings since the more fresh air being brought into the building from the outside, the less airflow is required per room to maintain the required freshness. In the example, line 3 returns the required ventilation rates for each zone.

The code uses the HAL semantic query interface to adjust the ventilation rates for their downstream rooms. We use a simple occupancy model based on time of day and class schedule obtained from a Google Calendar feed, and scale the ventilation rates accordingly. For example, in the morning, the ventilation rate is higher to accommodate the increased occupancy, while in the evening, the ventilation rate is reduced to save energy.

---

Figure 10: Execution flow of the personalized control application

The personalized control application: if its commands are overridden, it will immediately cause the control process to crash. However, this is difficult to do in traditional building control systems because separate control loops are in charge of the control process. The transaction manager reverts the blast action by undoing the changes to the building control system if the personalized control application and the error handler have been implemented using BOSS, providing evidence of varied execution flow.

---

We run the personalized control application concurrently with the HV AC optimization application. Since BOSS is a transactional system, reverting the room temperature takes time as the space slowly warms back up to steady state. Unlike traditional computer systems, reverting the room temperature takes direct occupant input to adjust room temperatures. The application writes to a room setpoint in response to a user request for cooling.

A second application, a personalized control system, has been implemented using BOSS, providing evidence of more complex, while showing how it helps applications to coexist.

---

We further evaluate BOSS in two ways: first, we examine how the system architecture makes implementing applications easier and more concise, while showing how it helps applications to coexist.

---

We use the auditing application to compute energy savings from the HV AC optimization and personal control applications responding to a user request for cooling. The personalized control application temporarily blast warm or cold air into the space in response to a user request for cooling. The application writes to a room setpoint in response to a user request for cooling.

---

Figure 10. Temperature drops while the cold blast is run and ventilation. One of its key features is the ability to take direct occupant input to adjust room temperatures. A second application, a personalized control system, has been implemented using BOSS, providing evidence of more complex applications responding to a user request for cooling.
More BOSS

- sMAP Hardware Presentation Layer
  - 30 Drivers, 30k data streams
- Archiver data storage service
  - 500 writes/sec
  - Stream cleaning and processing
- Family of apps
  - Personal ventilation and lighting control
  - Electric grid-aware consumption

<table>
<thead>
<tr>
<th>Name</th>
<th>Sensor Type</th>
<th>Access Method</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO Data</td>
<td>CAISO, NYISO, PJM, MISO, ERCOT</td>
<td>Web scrape</td>
<td>1211</td>
</tr>
<tr>
<td>ACme devices</td>
<td>Plug-load electric meter</td>
<td>Wireless 6lowpan mesh</td>
<td>344</td>
</tr>
<tr>
<td>EECS submetering project</td>
<td>Dent Instruments PowerScout 18 electric meters</td>
<td>Modbus</td>
<td>4644</td>
</tr>
<tr>
<td>EECS steam and condensate</td>
<td>Cadillac condensate; Central Station steam meter</td>
<td>Modbus/TCP</td>
<td>13</td>
</tr>
<tr>
<td>UC Berkeley submetering feeds</td>
<td>ION 6200, Obvius Aquisuite; PSL pQube, Veris Industries E30</td>
<td>Mosbus/Ethernet, HTTP</td>
<td>4269</td>
</tr>
<tr>
<td>Sutardja Dai, Brower Hall BMS</td>
<td>Siemens Apogee BMS, Legrand WattStopper, Johnson Control BMS</td>
<td>BACnet/IP</td>
<td>4064</td>
</tr>
<tr>
<td>UC Davis submetering feeds</td>
<td>Misc., Schneider Electric ION</td>
<td>OPC-DA</td>
<td>34 (+)</td>
</tr>
<tr>
<td>Weather feeds</td>
<td>Vaisala WXT520 rooftop weather station; Wunderground</td>
<td>SDI-12, LabJack/Modbus, web scrape</td>
<td>33</td>
</tr>
<tr>
<td>CBE PMP toolkit</td>
<td>Dust motes; New York Times BMS</td>
<td>CSV import; serial</td>
<td>874</td>
</tr>
</tbody>
</table>
Takeaways

• Applying computer systems design to buildings: lots of pieces, potential
  – Control systems
  – Mechanical systems
  – Occupants

• 30% electricity + steam savings, 60% lighting savings in test apps

• Many pieces at http://smap.cs.berkeley.edu

• Control systems + CS future work
  – Making use of the torrent of data?
  – Compile/enforce constraints into the network?
  – How to verify applications are behaving?
Thank you
proc = BossProcess(timeout=15min, auth_token=ABC)
while True:
    for dmp in hal.find('#OUT_AIR_DMP > #AH'):
        for vav in hal.find('#VAV < $\%s' % dmp.name):
            occ = model.estimate_occupancy(vav)
            vav.set_min_airflow((vav.min_fresh_air() / dmp.get_percent_open()) * occ)
    time.sleep(15*60)

Write applications in terms of relationship between hardware elements
THIS IS COMFORT-ON-DEMAND. IT CONSISTS OF TWO PARTS.

1. A fast-responding energy distribution system

   Chilled Sails: locally controlled radiant panels connected to building-wide hydronic mechanical system

   OLED Lights: locally controlled and ideally powered by on-site renewables

   Dedicated Outlet: powers down when you walk away and eliminates phantom load

   Heated Surface: (25-watt) capacitive-sensors direct energy to warm your hands (not your notepad)

   DC Fan: (17-watt) tunable controls and adjustable nozzles point air where you want it to go

   Heated Surface: (123-watt) pressure sensor triggers an adjustable heater to keep your feet cozy

2. A mobile control device

   Bluetooth: communicates your preferences
legacy solution: encode everything in point name

SDH.MEC-08.S5-01.AIR_VOLUME

building name
MEC: Modular Equipment Controller
S: VAV, 5: 5th floor, 01: the 1st one
quantity being measured

4/5/13  NSDI 2013: Lombard, IL
BOSS

1. Hardware presentation layer: sMAP
2. Hardware abstraction layer: device-specific logic
3. Time-series service: the archiver
4. Reliable control inputs: the transaction manager
5. Security: the authorization service

a collection of services enabling portable, robust applications for the physical environment

4/5/13 NSDI 2013: Lombard, IL
**writer 1** value: 69F

**writer 2** value: 73F

- No arbitration between applications
- Orphaned writes
Command Sequence

1. Set damper to 100% open
2. Set valve to 0% open
3. ... wait 10 minutes
4. Reset to “whatever was happening before”

What if…
1. #1 or #2 fail?
2. Client fails/becomes partitioned during #3?
3. Another application tries to do something?
BOSS solution: “transactions”

Extend writes with
- Priorities
- Leases
- Notifications
- Reversion sequences

overridden!

writer 1 value: 69F priority: 3 lease: 3600s

writer 2 value: 73F priority: 1 lease: 300s

<time passes>

writer 2 clear

writer 1 crashes

... writer 1 revert sequence runs