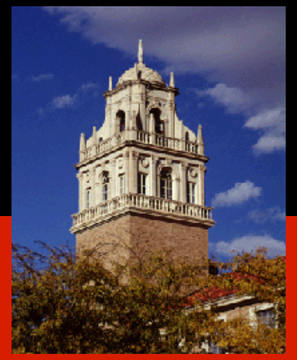


TEXAS TECH UNIVERSITY
Information Technology Division

High Performance Computing Center

Emulation of Automated Control of Large Data Centers At Scale Using Containers

Alan Sill

Senior Director, High Performance Computing Center, Texas Tech University

Co-Director, US National Science Foundation Cloud and Autonomic Computing Industry/University Cooperative Research Center

Workshop on Autonomic Management of Large-scale Container-based Systems
ICCAC 2017: International Conference on Cloud and Autonomic Computing

Sep. 18, 2017

Research Group For Work Cited:

Faculty members:

Alan Sill, High Performance Computing Center, Texas Tech University

Yong Chen, Associate Professor, Dept. of Computer Science, Texas Tech University

Dong Dai, Research Assistant Professor, Dept. of Computer Science, Texas Tech University

Graduate students:

Ali Nosrati, MS student (containers and scaling framework)

Elham Hojati, PhD student (benchmarking and scaling measurement)

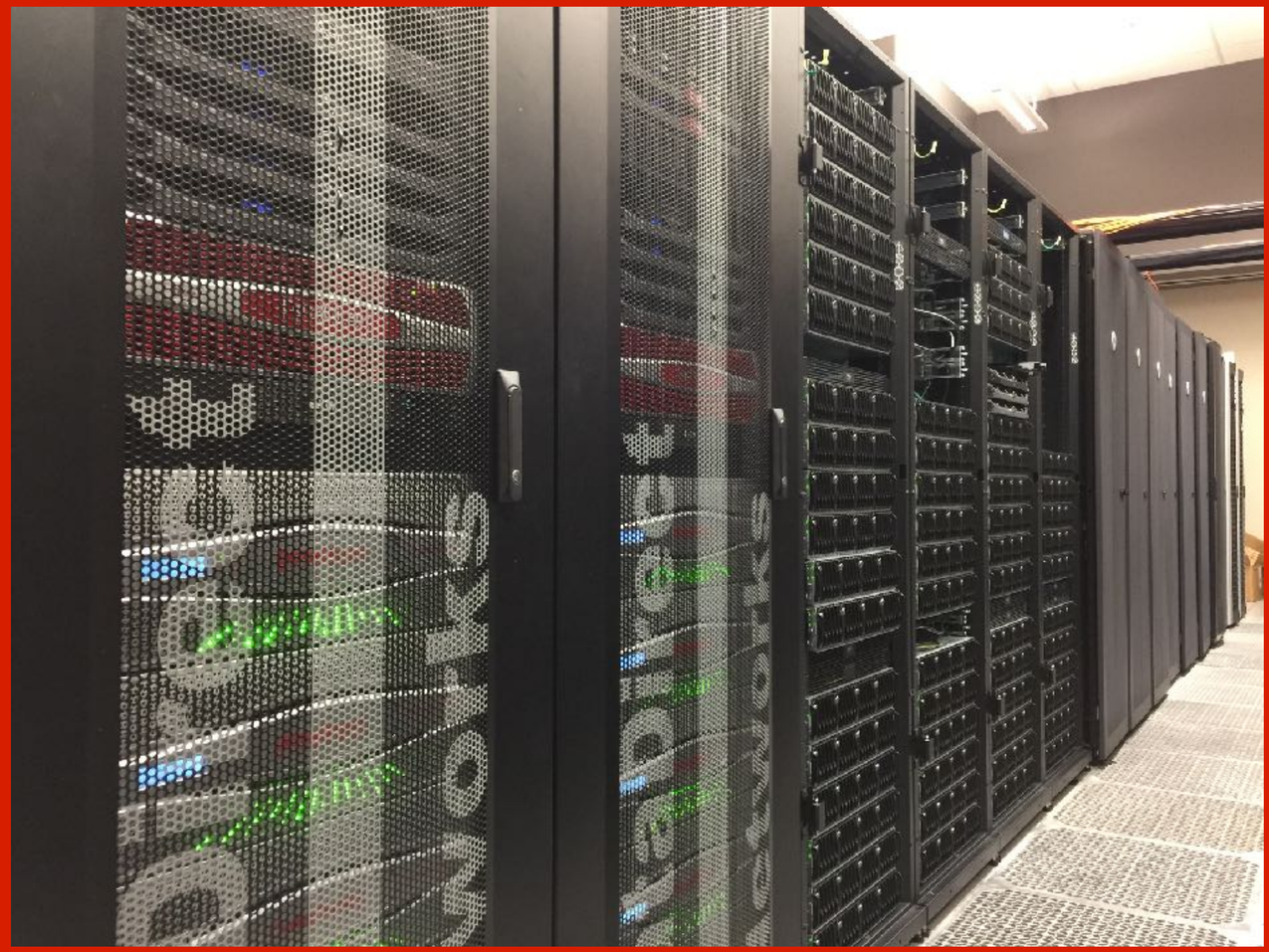
Ghazanfar Ali, PhD student (protocols and design comparisons)

Priyanka Kumari, MS* (specification coverage and assertion testing)

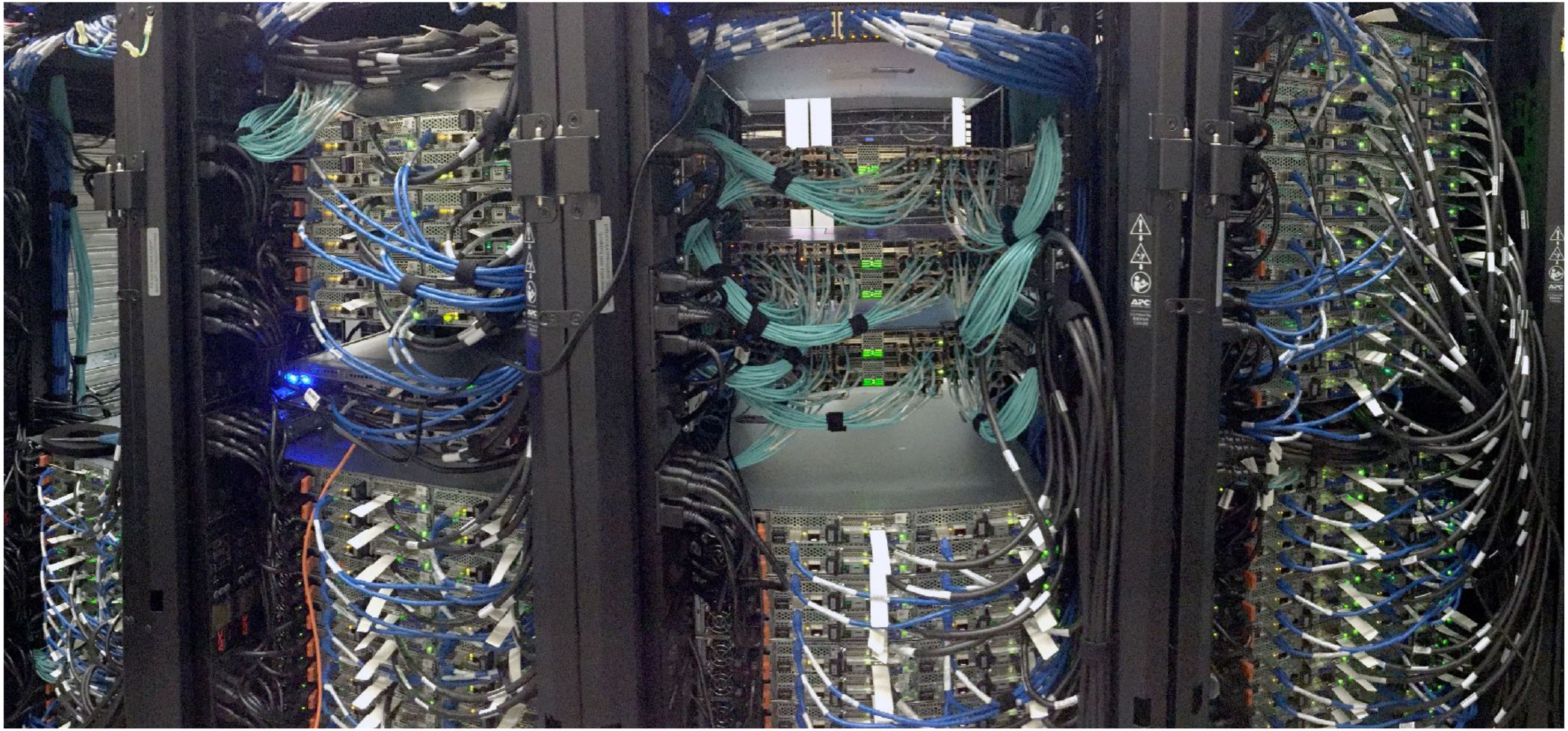
* (graduated)

Outline of this talk

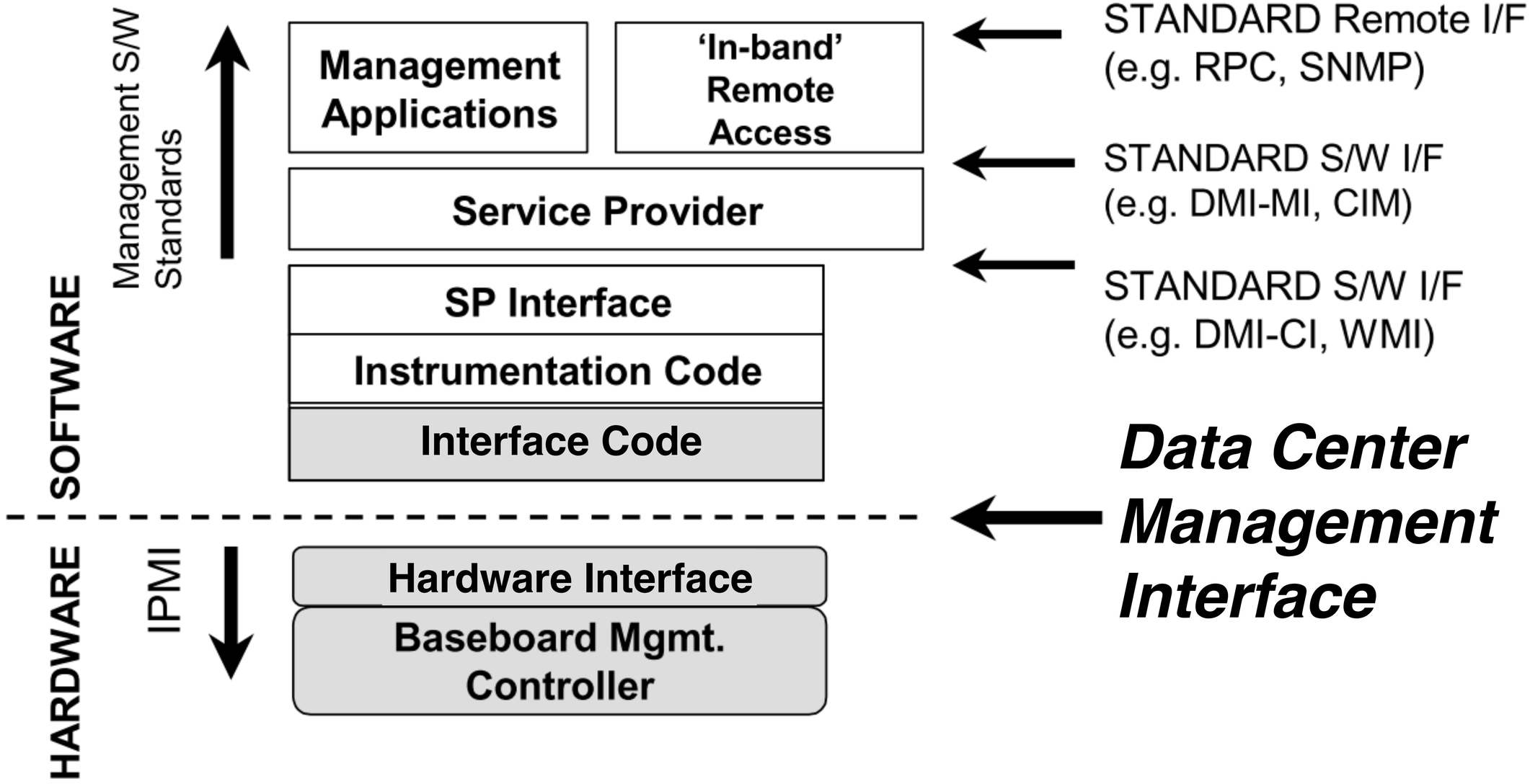
- Modern data center control
  - Description of the problem
  - Past and new current standards
  - Testing tools
- Scaling the environment
  - Testing with real servers
  - Testing with containers
- Conclusions and future work



Modern Data Center Control

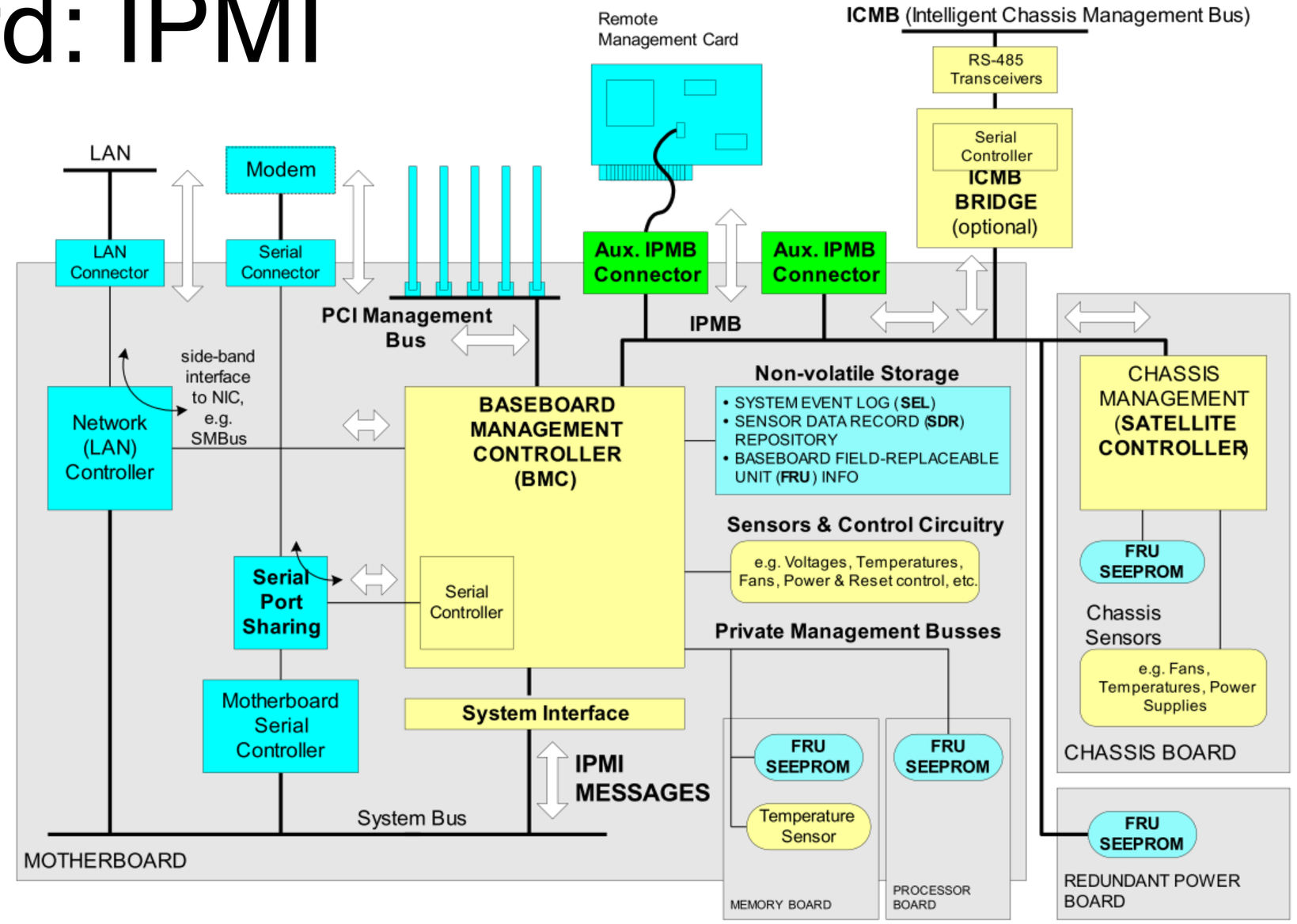


Modern Data Center Control: Description of the Problem



Past Standard: IPMI

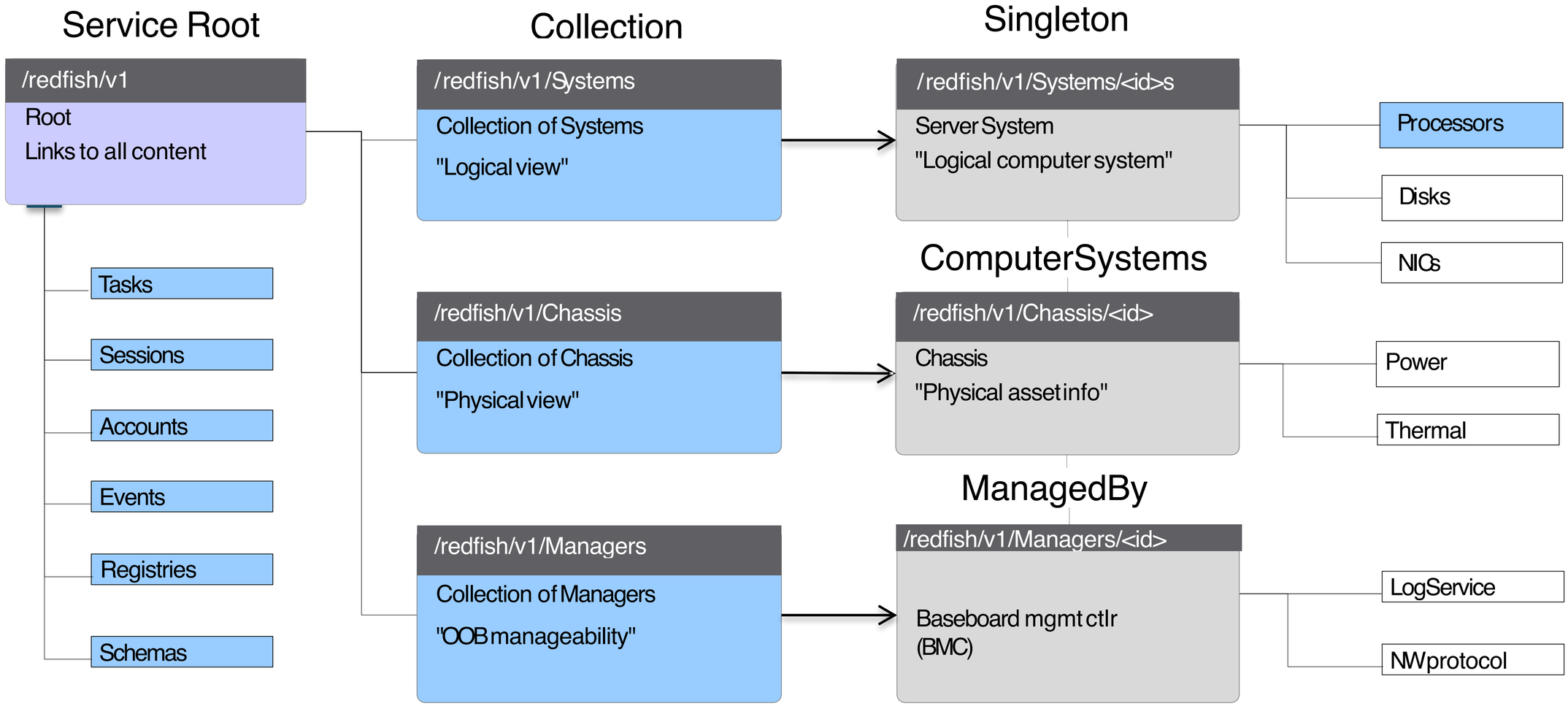
Up to now, management of the Baseboard Management Controller (BMC) has been the province of the Intelligent Management Platform Interface (IPMI), built into many items of server-class hardware from many different manufacturers.



. IPMI specification, v2.0. http://www.intel.com/content/www/us/en/servers/ipmi/ipmi- second-gen-interface-spec-v2-rev1-1.html

New Standard: ***Redfish***

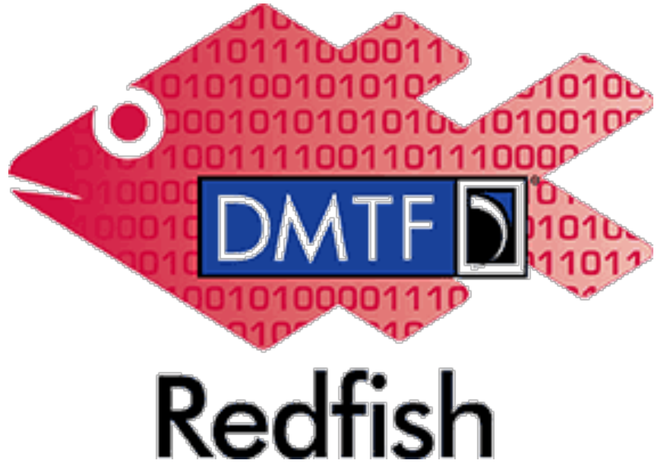
Developed by the Distributed Management Task Force (DMTF) as a modern replacement for IPMI.



http://www.dmtf.org/sites/default/files/D2%20T1%20S6%20Managing%20Servers%20with%20Redfish_v3.pdf

Initial Goals of Redfish Standard (2014-2015)

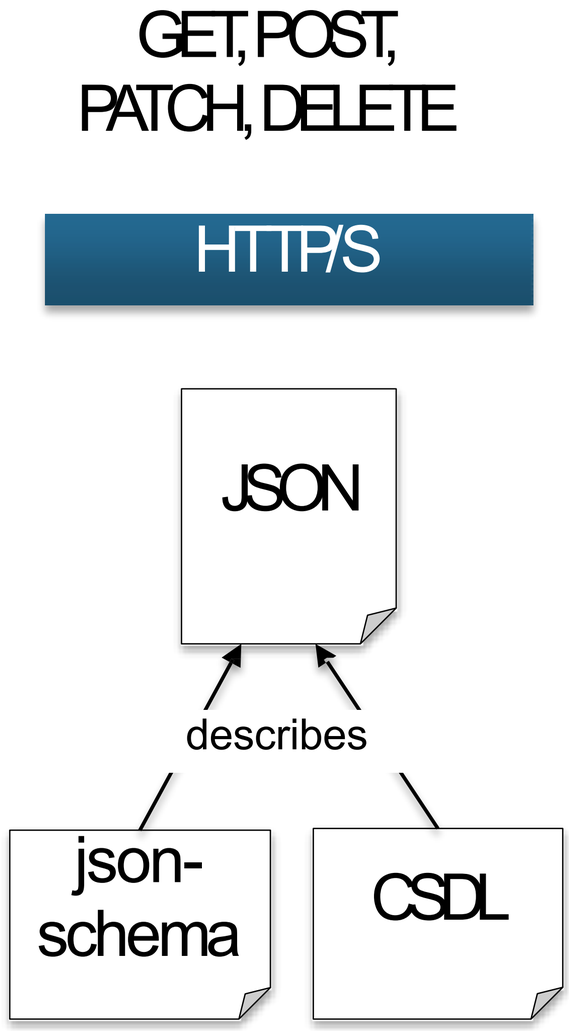
- RESTful interface over HTTPS (modern paradigm)
- Leverages existing Internet standards and tool chains
- Deployable on existing management controllers
- Easy-to-use data model (JSON format based on OData)
- Usable by client applications and browser-based GUIs
- A secure, multi-node capable replacement for IPMI-over-LAN
- Schema-backed but human-readable output
- Usable by beginners and experts, and multiple partners
- Cover popular use cases and data center requirements
- Applicable to open-source projects (e.g., intended to meet OCP Remote Machine Management requirements)



http://www.dmtf.org/sites/default/files/D2%20T1%20S6%20Managing%20Servers%20with%20Redfish_v3.pdf

Redfish v1 Protocol Suite Features

- RESTful API architectures are rapidly replacing SOAP for general-use web services
- HTTP/HTTPS protocol for a request and response mechanism, with alert subscription
- Data encapsulated in JSON, and adheres to a data schema
- Data schema is expressed in json-schema and OData CSDL[1]
- Device discovery using SSDP/uPnP (optional)



[1]OData is an OASIS Standard
CSDL = Common Schema Definition Language

http://www.dmtf.org/sites/default/files/D2%20T1%20S6%20Managing%20Servers%20with%20Redfish_v3.pdf

Initial Server Control Goals: DMTF Redfish v1

- **Retrieve "IPMI class" data**
  - Basic server identification and asset info
  - Health state
  - Temperature sensors and fans
  - Power supply, consumption and thresholds
- **Discovery**
  - Service endpoint (network-based discovery)
  - System topology (rack/chassis/server/node)
- **Basic I/O infrastructure data**
  - Host NIC MAC address(es) for LOM devices
  - Simple hard drive status / fault reporting
- **Security**
  - Session-based, leverages HTTPS
- **Perform Common Actions**
  - Change boot order / device
  - Reboot / power cycle server
  - Set power thresholds
- **Access and Notification**
  - Serial console access via SSH
  - Event notification method(s)
  - Logging method(s)
- **BMC infrastructure**
  - View / configure BMC network settings
  - Manage local BMC user accounts
- ***… but Redfish scope is still growing!***
  - (See next page)

http://www.dmtf.org/sites/default/files/D2%20T1%20S6%20Managing%20Servers%20with%20Redfish_v3.pdf

Moving Beyond Redfish v1

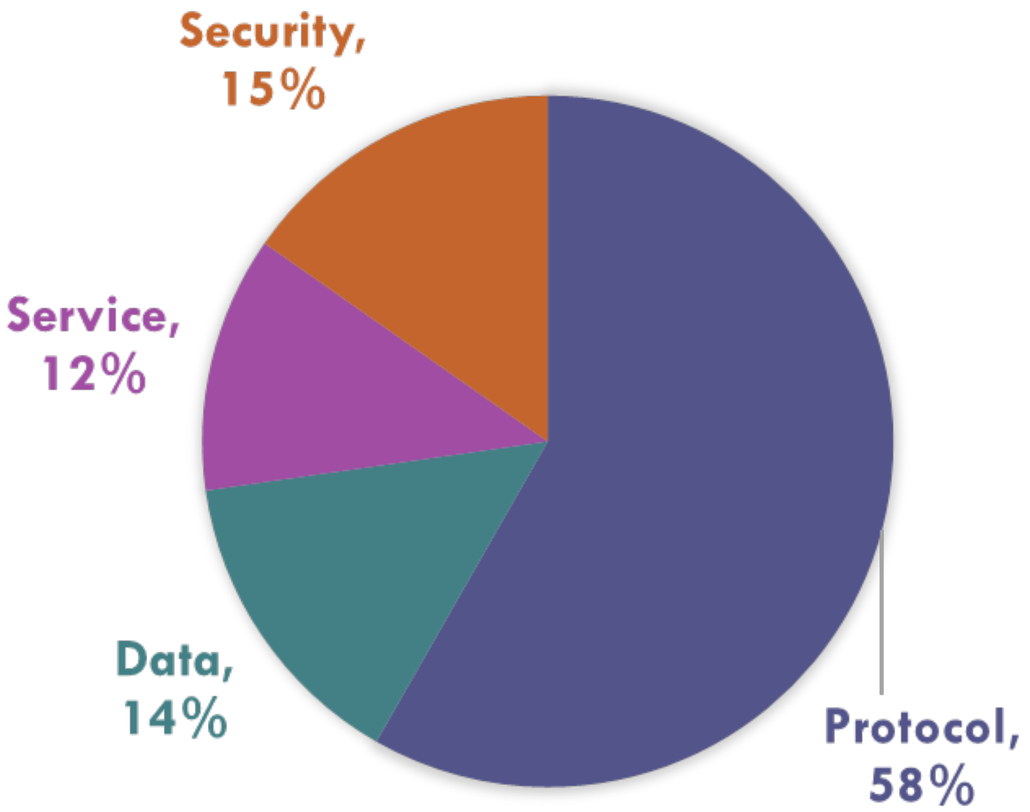
- **Version 1 was focused on Servers**
  - Intended to represent full server category: rackmount, blades, HPC, rackscale, future
- ***Future is happening now!***
  - Task forces created over time in response to sufficient need or interest
  - Additional features coming out approximately every 4 months
  - Models released for BIOS, disk drives, memory, storage, volume (2016.1); endpoint, fabric, switch, PCIe device, zone, software/firmware inventory & update (2016.2), advanced communications devices (multi-function NICs), host interface (KCS replacement), privilege mapping (2016.3), and composability (2017.1)
  - Client, validation, mockup, simulation, emulation, validation, and conformance tools available and being developed.
- **Developer hub: redfish.dmtf.org**
  - Schema Index, specifications, registries
  - GitHub for Redfish Tools
  - Mockups, user forum, videos, documentation, white papers, tools mentioned above, & more.
- **Also expanding scope over time to more of IT infrastructure**
  - Working with SNIA to cover more advanced Storage (Swordfish released Aug. 2016)
  - Working with The Green Grid to cover Facilities (Power/Cooling)
  - Work with the IETF to cover some level of Ethernet Switching and to create a programmatic Redfish mapping
  - Work registers with UEFI Forum and OCP
  - Mockups include proposed OCP Redfish Profile

More detail: http://dmtf.org/standards/redfish

Verifying Redfish Service Conformance

P. Kumari, Master's thesis, 2017

- Service conformance tool built by TTU extending initial work of Intel for DMTF
- Verifies the conformance of a Redfish service to assertions extracted from the Redfish specification
- Includes a total of 242 "shall" assertions so far (initial version)
- Sections include
  - Protocol testing: 141
  - Data model testing: 35
  - Service details testing: 29
  - Security testing: 37



Redfish assertion categories

https://github.com/DMTF/Redfish-Service-Conformance-Check

Redfish Service Conformance Checker

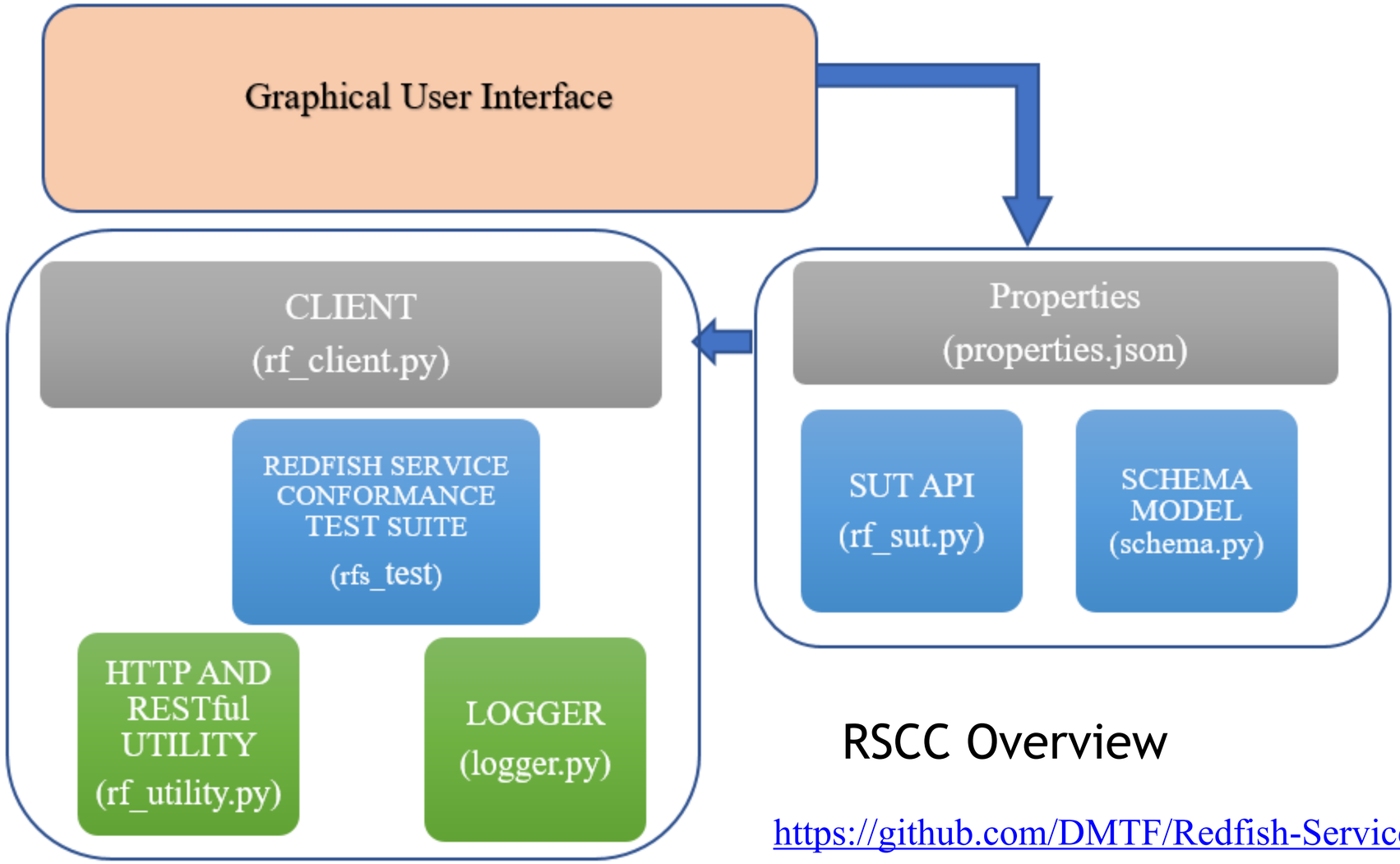
P. Kumari, Master's thesis, 2017

- Checks an operational Redfish Service to see that it conforms to the normative statements from the Redfish specification.
- Example:

Section 6.4.2.4.2 :Retrieving Collections	Rule	Rule	Comments
Clients shall not make assumptions about the URIs for the resource members of a collection.	Yes	No	
Retrieved collections shall always include the count property to specify the total number of members in the collection.	No	Yes	Assertion Tested and Passed

Overview of service conformance check

P. Kumari, Master's thesis, 2017

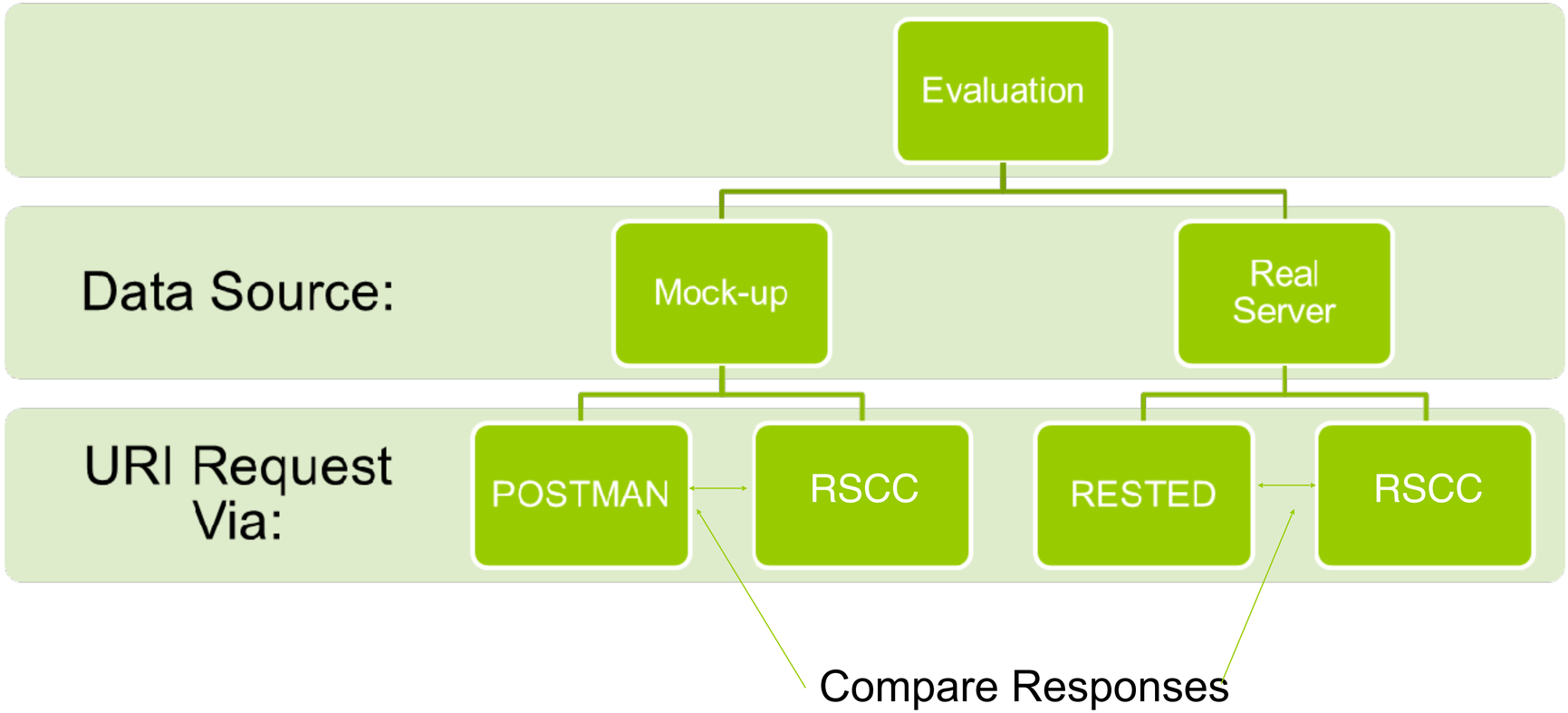


RSCC Overview

https://github.com/DMTF/Redfish-Service-Conformance-Check

Evaluation - RSCC

P. Kumari, Master's thesis, 2017



- Regression Testing of the Tool:
  - Some requests were not valid on Mock-up data, they had to be tested on Real Server

Scaling the Environment

Scalability benchmarking is for measuring the performance of a system for a sequence of workload runs in which the load or/and capacity or/and setting of the system are changed between runs of workloads [1].

Elasticity benchmarking is for measuring the performance of a system for a workload in a way that the load or/and capacity or/and setting of the system are changed in the middle of running the workload [1].

Other quantities and characteristics of interest:

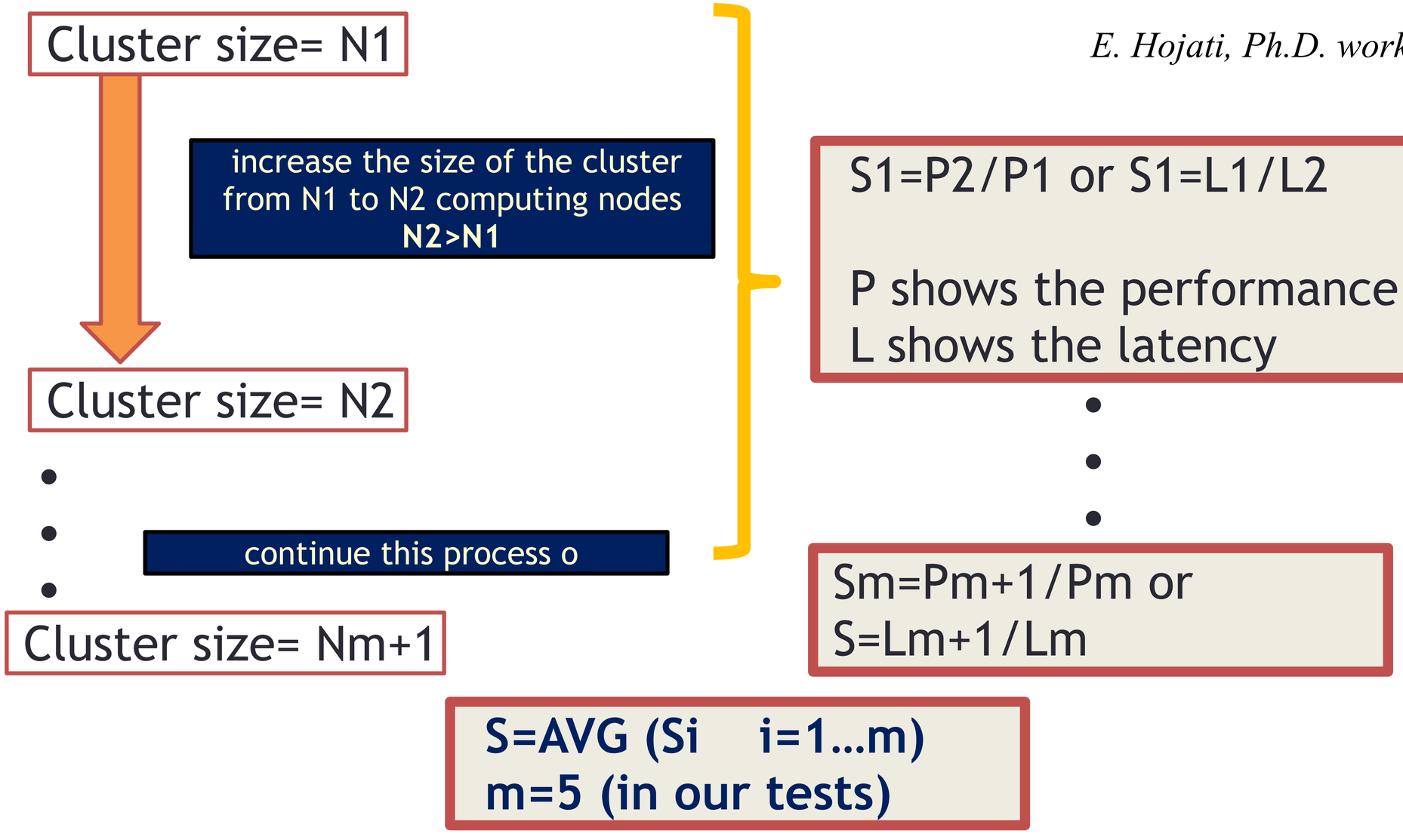
- Latency
- Security
- Reliability
- Simplicity of usage

E. Hojati, Ph.D. work (in progress)
G. Ali, Ph.D. work (in progress)

[1] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, "Benchmarking cloud serving systems with YCSB", SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing, Pages 143-154

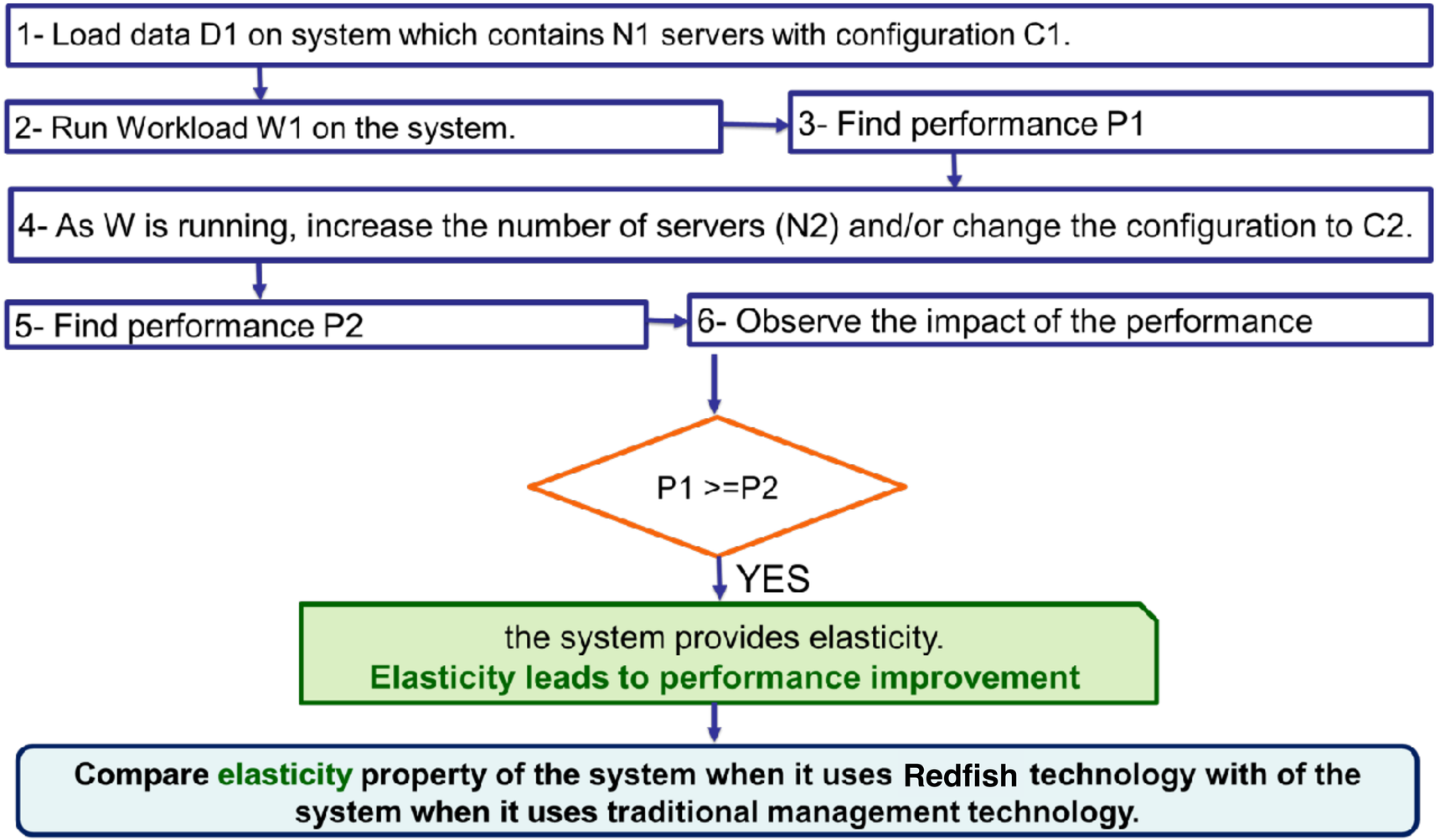
DEFINED SCALABILITY METRIC

E. Hojati, Ph.D. work (in progress)



Plan of Benchmarking Scale-up Scenarios

E. Hojati, Ph.D. work (in progress)



Example Testing Workloads

E. Hojati, Ph.D. work (in progress)

W1: Temperature checking
W2: Fan Speed checking
W3: Sensor Data Record (SDR) checking
W4: Chassis Status checking
W5: System Event Log (SEL) checking
W6: Field Replaceable Unit (FRU) information checking
W7: Baseboard Management Controller (BMC) User account information checking

Example commands:

IPMI: ipmitool -I lanplus -H 10.101.$j.$i -U *account* -P *password* sensor reading Temp

Redfish:
python3.4 Redfishtool.py -r 10.101.$j.$i -SAlways -u *account* -p *password* -ss -vvv Chassis -1 Thermal -P Temperatures

Redfish “raw mode”:
python3.4 Redfishtool.py -r 10.101.$j.$i -SAlways -u *account* -p *password* raw GET https://10.101.19.$i/Redfish/v1/Chassis/System.Embedded.1/Thermal/

Real Cluster Testing

Quanah Cluster:

- **224 Dell™ compute nodes**
  - 36 CPU cores: Intel(R) Xeon(R) CPU E5410 @2.33GHz
  - 192 GB of RAM

- **Operating System:**
  - CentOS 7.2.1511, 64-bit version, Kernel 3.10

- **Management Network:**
  - Ethernet, 1Gbps

- **Connection Network:**
  - Intel ™ Omnipath , 100Gbps

- **Location**

  Texas Tech University (Lubbock, TX)

  High Performance Computing Center

Scalability Test: Full Tree Traversal (Real Cluster)

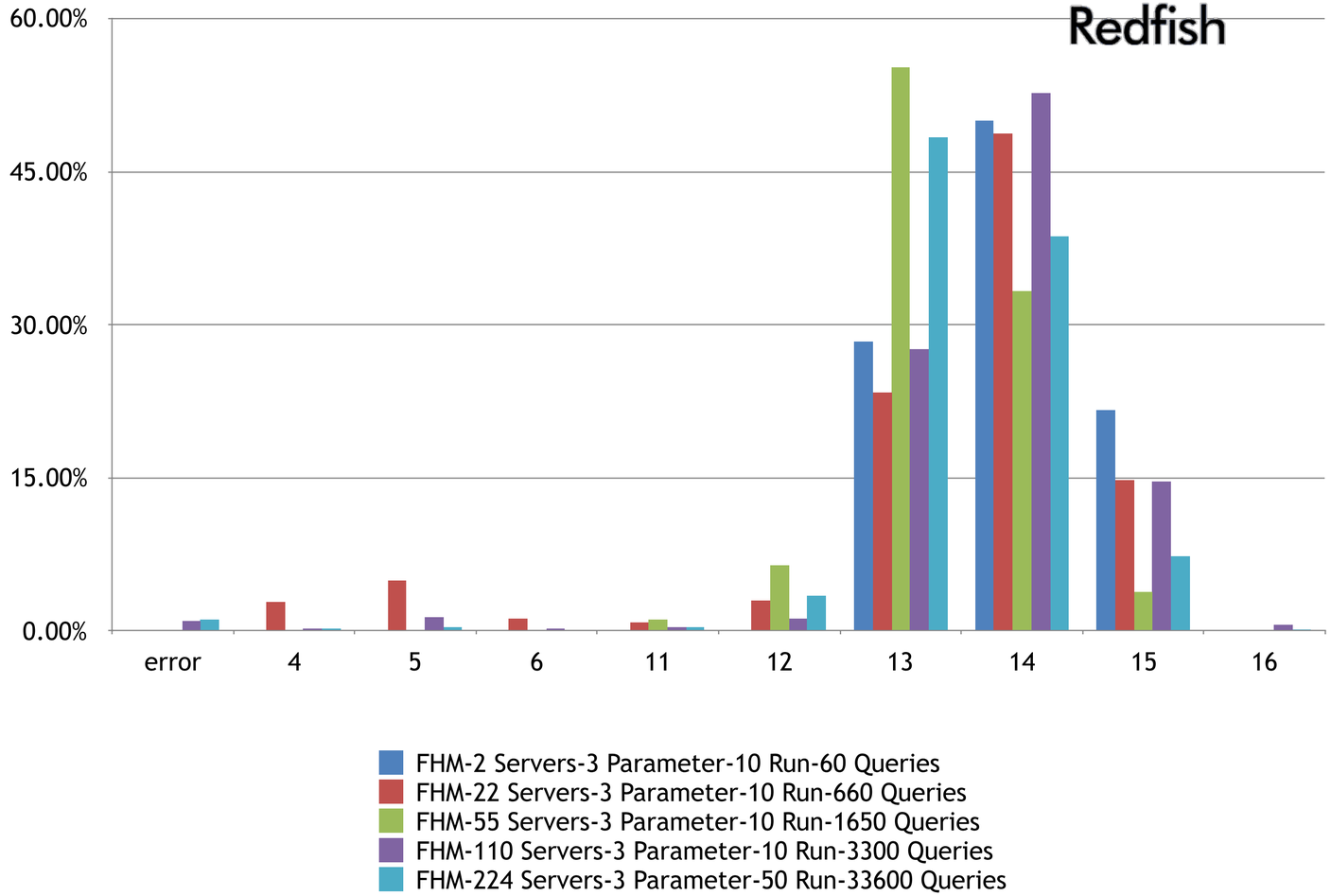
A. Nosrati, MS work (in progress)
E. Hojati, Ph.D. work (in progress)

- **Full HyperMedia query**
- **10 or 50 query against each node**

For each piece of information (3 Parameters), including:

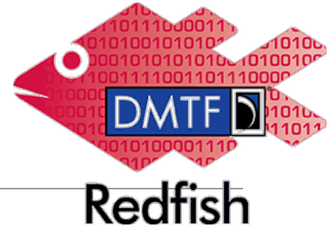
- ✓ Fan speed
- ✓ Power usage
- ✓ Temperature

- **Different number of nodes**
  - ✓ 2 nodes
  - ✓ 22 nodes
  - ✓ 55 nodes
  - ✓ 110 nodes
  - ✓ 224 nodes



Scalability Test: “Raw” Specific Directed Command (Real Cluster)

A. Nosrati, MS work (in progress)
E. Hojati, Ph.D. work (in progress)

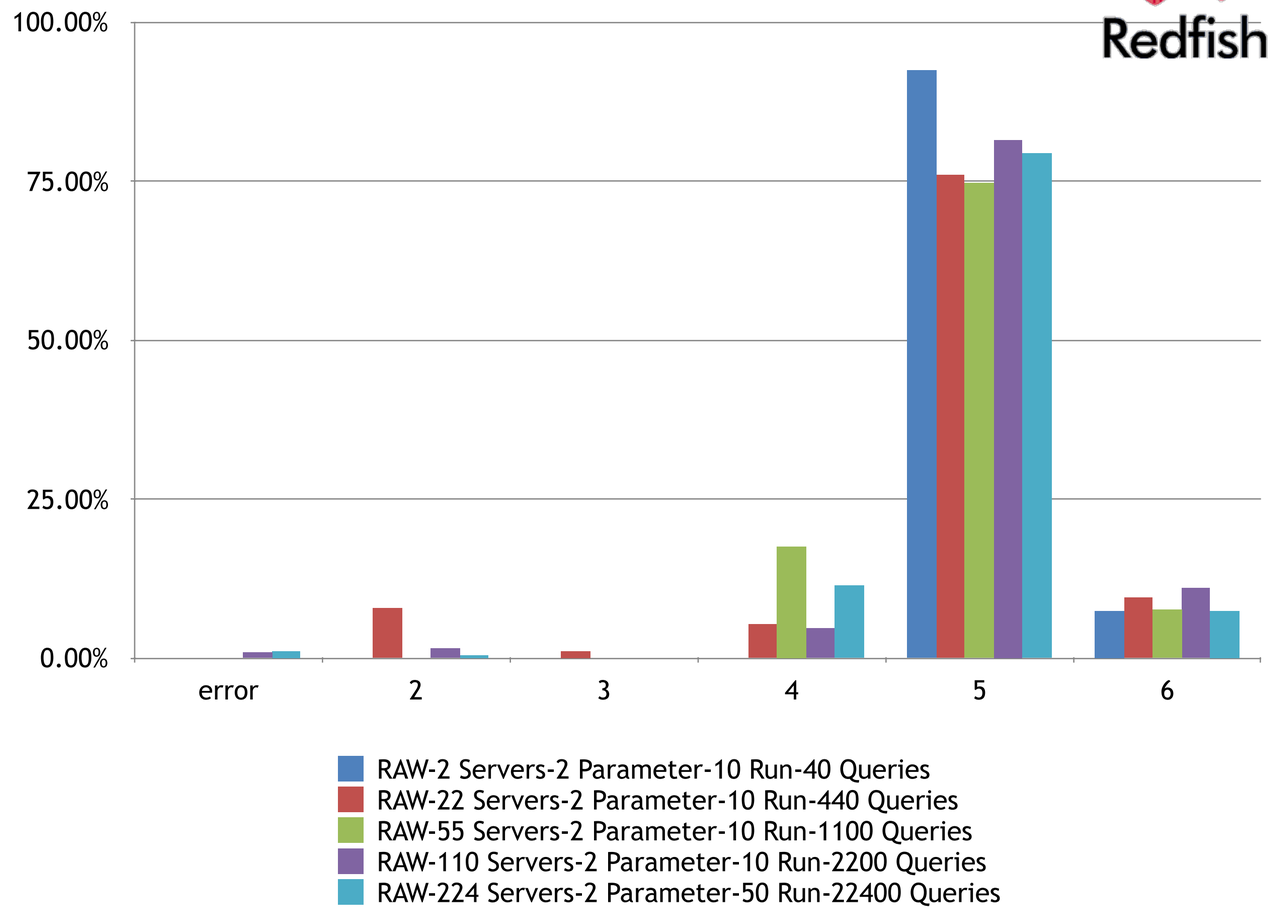


- **RAW query**
- **10 or 50 query against each node**

For each piece of information (3 Parameters), including:

- ✓ Fan speed
- ✓ Power usage
- ✓ Temperature

- **Different number of nodes**
  - ✓ 2 nodes
  - ✓ 22 nodes
  - ✓ 55 nodes
  - ✓ 110 nodes
  - ✓ 224 nodes



Initial Results - Real Server Testing

E. Hojati, Ph.D. work (in progress)

(Here I summarize conclusions only - for full details see papers and posters submitted to the Utility and Cloud Computing 2017 conference (UCC 2017) and its associated workshops.)

	Latency	Scalability	Reliability	Security	Simplicity of usage
IPMI	✓✓	✓	×	×	×
Redfish	× (**)	✓✓	✓	✓	✓
Redfish-raw	✓	✓✓✓	✓	✓	-

*(** - Initial query only. Latency diminishes as a factor in multiple simultaneous queries)*

Results show UDP features of IPMI are a disadvantage for large numbers of servers at high rates. By contrast, TCP nature of Redfish queries allows simultaneous queries of large numbers of endpoints in a scatter/gather fashion, ultimately allowing much larger numbers of queries to be issued at a time with higher reliability.

Some interesting findings

A. Nosrati, MS work (in progress)

- Running Full HyperMedia query (full Redfish schema traversal) against servers is **significantly longer** than running raw query with final URI

- In large data centers, the client that is monitoring many nodes needs to poll nodes' data in parallel (**Multi thread**)

The time it takes to poll all nodes can approach the time to poll one node

(Through the client with **sufficient performance**)

Which shows that

Redfish protocol –even running on BMCs **with modest response times– can be used to monitor large DC**

- Nearly all the compute nodes respond at a small range, not at a wide range:

✓ Full HyperMedia: **13s, 14s, 15s**
✓ Raw: **4s , 5s, 6s**

Scaling to Large Numbers of Endpoints Using Containers

Baseboard management controller (BMC) features can be emulated in a relatively small amount of logic, making it feasible to simulate all relevant operations of a large data center by replicating a large number of independent, fully functional copies in a scaleable framework.

This is a good match to the general features of containers!

Need to solve the problems of creating a software-defined network that is realistic enough to match the characteristics found in data centers while incorporating measured response times of real servers, including variability in response.

Started out using Docker with Swarm networking for initial tests. Reached >5000 servers emulated in a small half-rack cluster of old machines.

With modest numbers of current-model machines, ***it is possible to simulate realistic behavior of many tens of thousands of machines easily.*** Problem becomes how to handle container networking for even larger numbers (100s of 1000s of emulated BMCs).

Emulated Redfish™ Servers

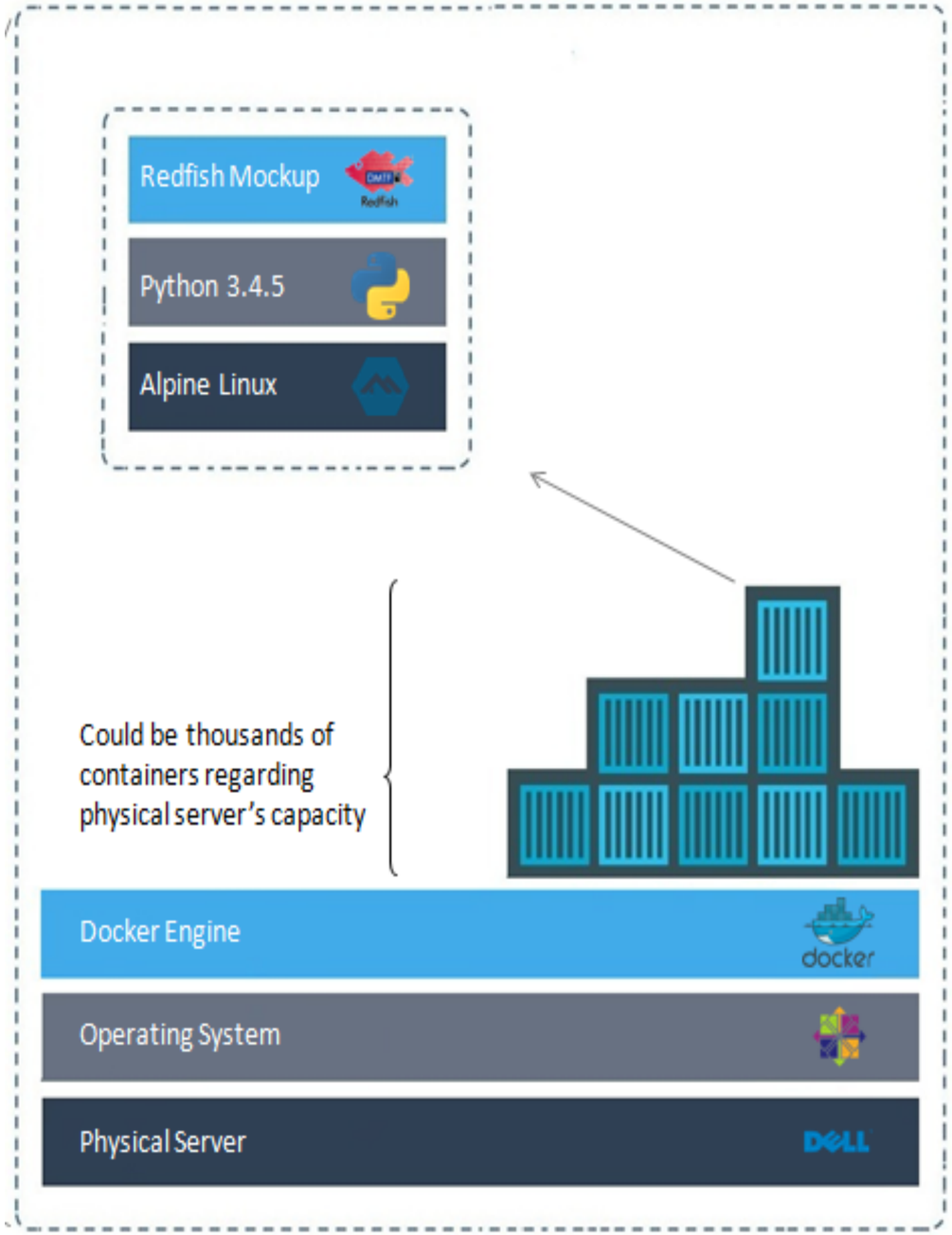
Docker image (**85 MB**) including:

- Alpine Linux
- Python 3.4
- Redfish Mockup Server 0.9.3

Redfish Mockup Server

Python code that uses mockup data obtained from theoretical or real servers and serves Redfish responses on a specified IP/Port

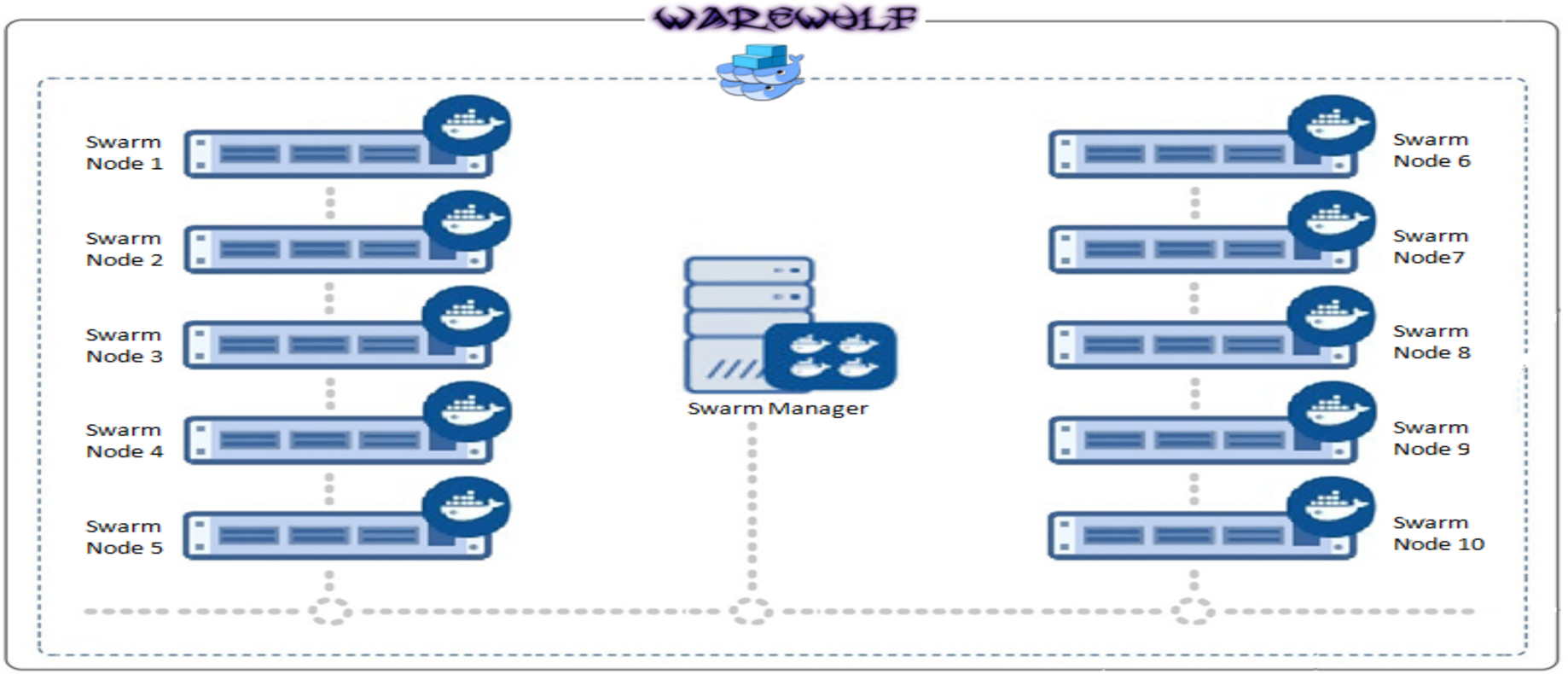
- **Different type of servers can emulated** (Responses through **mockup creator**)
- **Adjustable response time**



Simulated data center

A. Nosrati, MS work (in progress)

- **Warewulf:** System provisioning (Centos 7.2 on all nodes)
- **Docker-machine:** Docker engine installation and setup (Docker 1.13 on all nodes)
- **Docker Engine:** Support containerization
- **Docker swarm:** Provides native clustering capabilities (for scaling out an application)



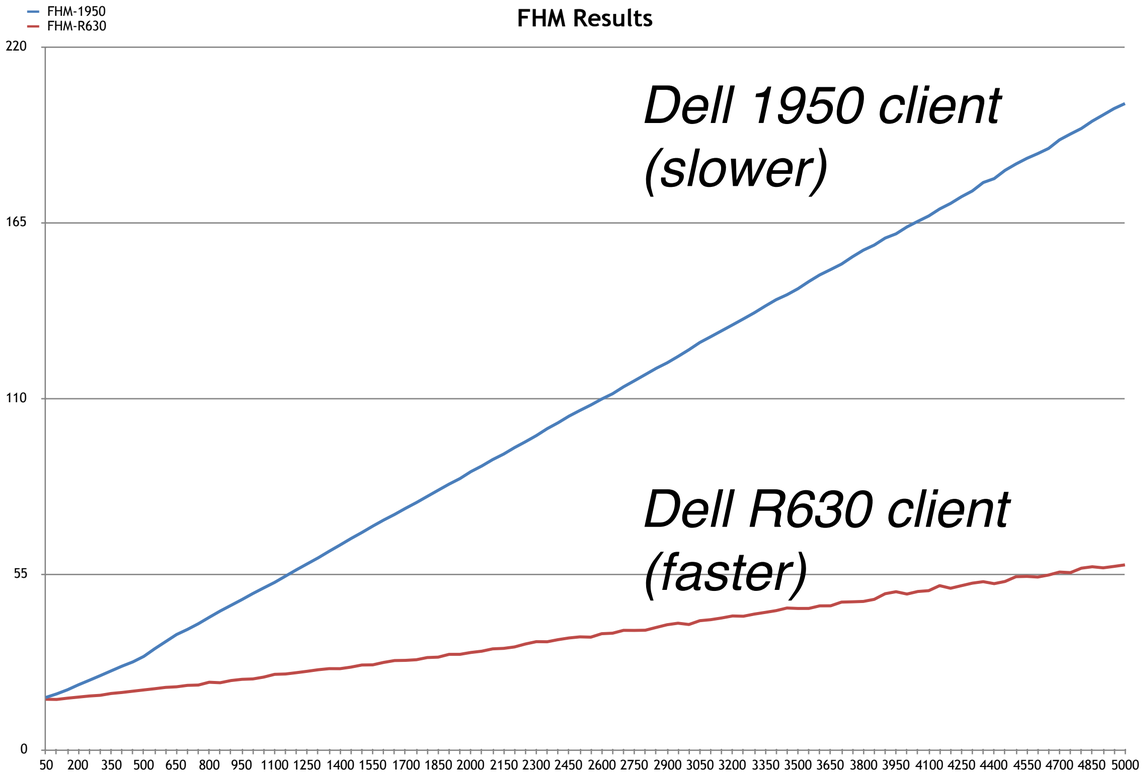
Test bed cluster

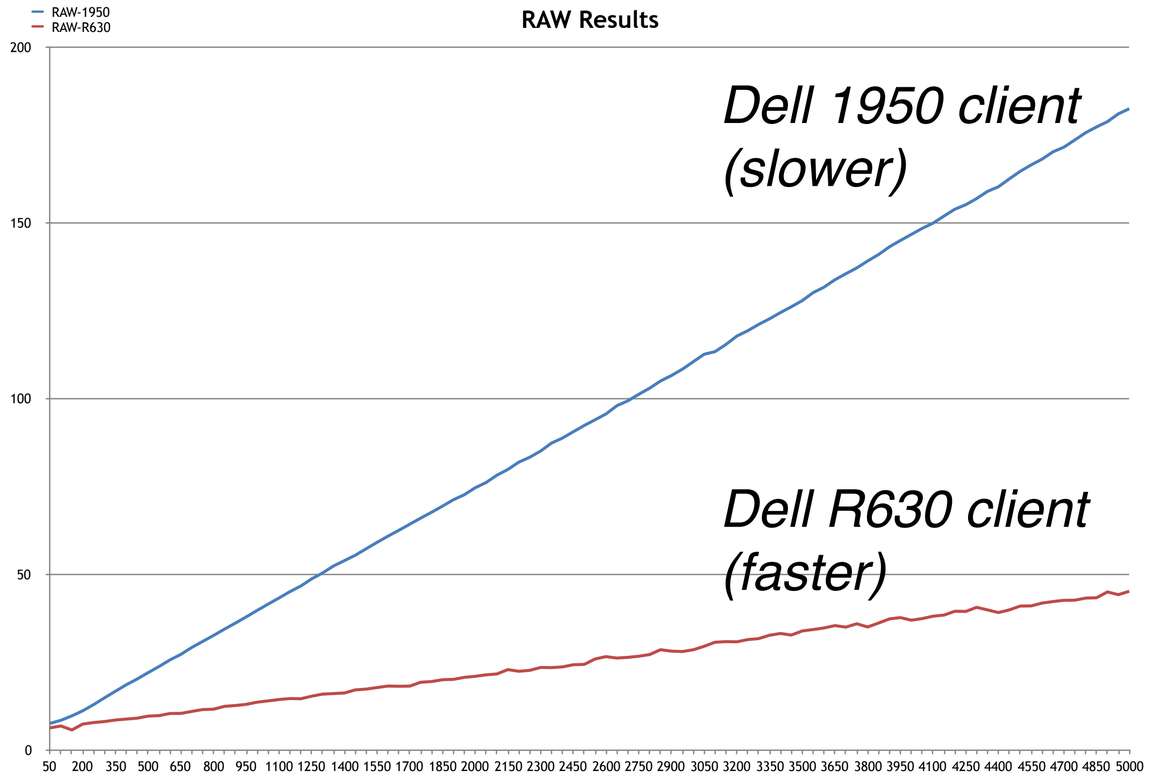
- 18 nodes (Dell POWEREDGE 1950)
  - 1 Docker swarm manager
  - 17 Docker swarm nodes
  - 72 CPU cores: Intel(R) Xeon(R) CPU E5410 @2.33GHz
  - 288 GB of RAM
  - 2.9 TB HDD
  - Network: 1 Gbps BMC, 2*1Gbps Ethernet (each node)
- Operating System: CentOS 7.2.1511, 64-bit version, Kernel 3.1
- Able to serve up to 5000 copies of the emulated Redfish server simultaneously!

Client Becomes Bottleneck!

A. Nosrati, MS work (in progress)

Results with containers tuned to match realistic behavior of servers show that time to query large number of emulated machines grows linearly with number queried, and becomes faster with faster client machines. Observation of load on TCP client generating the queries shows that it, and not the BMCs or network, is the source of the limitation.





Conclusions and Future Work

Results to date show that the TCP/IP nature of the new Redfish standard has many advantages and a few notable consequences in terms of overall client and server design:

1. Collisions due to UDP nature of IPMI can be avoided in Redfish queries
2. As a consequence, scatter/gather and direct queries can be used with confidence

Client becomes the bottleneck in large numbers of simultaneous queries and commands.

Future work is going on in our group to study the network packet and wire protocol implications of these findings. Meanwhile, newer generations of BMCs and related data center equipment are sure to be faster and open up the possibility to address, control, and monitor very large quantities of endpoints at a time with timescales that are much faster than with the previous generation of data center control.

Containers have been proven to be capable of scaling to simulate data centers of up to hundreds of thousands of endpoints with reasonable sizes of simulation servers.

Conclusions and Future Work (cont'd)

To go beyond the limitations of the Docker Swarm approach described here, we already have work in progress using the newly released IPVLAN methods to define and control much larger software-defined networks.

In combination with the previously described methods, we hope to be able to simulate very large data centers from the point of view of their control and command features with modestly-sized simulators and use these simulations to drive design of a new class of data center management hardware and software.

Full credit for this work should go to the graduate students cited, with thanks also to our industry partners in DMTF, as well as to Dell Inc. for loan of equipment used to carry out many of these tests.