Introduction to "ivy"

Block storage synthetic workload generator with real time dynamic feedback control

Hitachi Data Systems

2016-05-16

Allart Ian Vogelesang ian.vogelesang@hds.com +1 408 396 6511

Why ivy?

HITACHI
Inspire the Next

- Reduce cost
- Speed time to market
- Improve quality
- Simplify test setup
- Measure IOPS at a specified response time (dynamic feedback control)
- Get trust in measurement results through open source

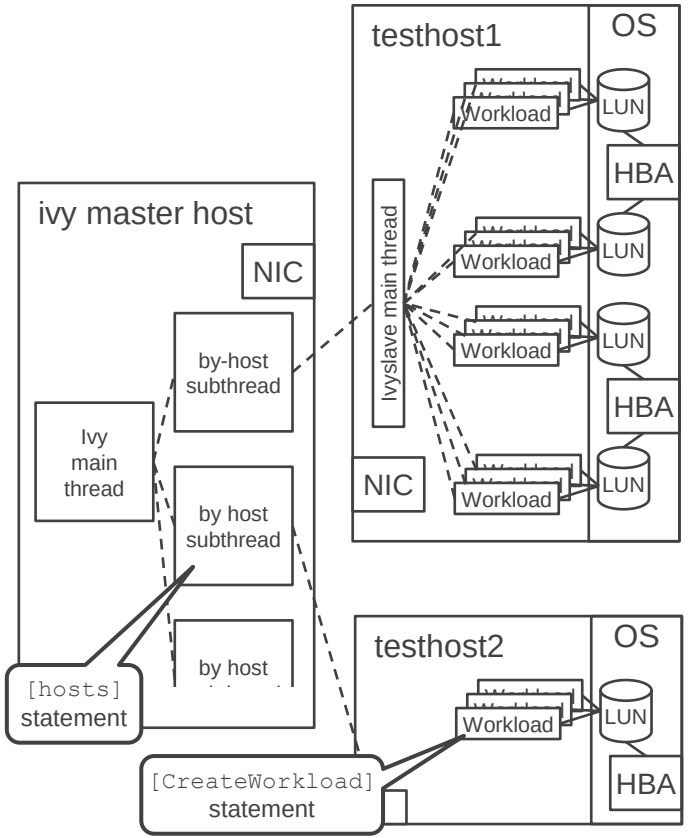
ivy in one chart

Items in italics only available to authorized Hitachi internal users using command device connector and license key

HITACHI
Inspire the Next

- Written in C++ using Linux native kernel Asynchronous I/O interface
  - Lowest CPU overhead – minimizes test host hardware cost & maximizes measurement accuracy
  - Designed for open source / vendor independence*, with a separately packaged Hitachi proprietary component.*
- Super easy test setup – select by LDEV, by Port, by PG, Pool ID *(by drive type with command device)*
- Can test for minimum time necessary to make a valid measurement to a specified +/- accuracy
- Real-time dynamic feedback control (DFC) enables for the first time to
  - "measure IOPS at 1 ms service time" (using DFC on rolled up host workload data – any vendor's equipment)
  - *"measure IOPS at 50% owning MP core % busy" (using DFC on filtered & rolled up command device data)*
- *Records subsystem command device performance data synchronized with host workload data*
  - *Makes validating / calibrating models easier*
- Records only valid measurement data
  - Checks number of resources reporting (ports, PGs, LDEVs, etc.) and that none running too slowly. *Checks that subsystem does not have any failed components (with command device).*
- Enables workflow automation using ivyscript programming language wrapper around ivy engine.

Scalable



- In ivy, you can layer multiple named “workload” I/O generator threads on LUNs
- Each workload thread uses an I/O sequencer plug-in that generates I/Os in scheduled I/O start time sequence.
  - We try to keep a few I/Os “pre-computed” and ready to go at all times.
  - Generalized interface allows use of any I/O pattern generator, e.g. synthetic or playback.
- Use of C++ with Linux kernel Asynchronous I/O interface means each workload thread can drive any queue depth up to limits of the OS / HBA with state of the art minimal CPU overhead.

ivy – designed for vendor independence

HITACHI
Inspire the Next

1. Vendor proprietary simple SCSI Inquiry-based LUN discovery tool
   - Vendor's attribute names / values become selectable in ivy.
   - Hitachi LUN discovery tool is open source - https://github.com/Hitachi-Data-Systems/LUN_discovery

2. Ivy
   - Open source - https://github.com/Hitachi-Data-Systems/ivy
   - HDS Community site - https://community.hds.com/groups/ivy
     - (Send Ian an email to get an invitation to join, but will be public very soon.)

3. For authorized internal Hitachi users with code and license key - vendor proprietary "connector" to retrieve real time storage product configuration and/or performance data
   - Enables ivy dynamic feedback control on internal storage metrics.

ivy - open source / vendor independent

HITACHI
Inspire the Next

- To support any vendor's architecture and terminology in ivy, all you need is a LUN discovery tool that makes a csv file with
  - a header line that defines vendor-specific attribute names, and
  - a data line for each LUN decoding that LUN attributes.

- Hitachi's `LUN_discovery` tool set using "`showluns.sh`" script produces something like this:

```
host,LUN name,HDS product,port,LDEV,PG,CLPR, …
testy1,/dev/sdxy,VSP,1A,00:00,1-1,CLPR0, …
testy2,/dev/sdyz,VSP,2A,00:01,1-2,CLPR0, …
```

Functionality expands with proprietary connector

HITACHI
Inspire the Next

- Vendors are encouraged to prepare their own proprietary "subsystem connector" interface tools to be used with ivy.
  - Collect subsystem configuration data to augment SCSI Inquiry attribute data
    - Select a test configuration based on attributes such as disk drive type.
  - Collect real time subsystem performance data, synchronized and aligned with test host workload data.
    - Facilitates development of modeling tools.
  - Use dynamic feedback control to find the IOPS to reach a target value for
    - subsystem MP % busy, or
    - drive % busy, or
    - cache dirty data % full, etc.

Hitachi's proprietary connector does all those things

HITACHI
Inspire the Next

- Hitachi's "ivy command device connector" (ivy_cmddev) is not part of the ivy open source project and will remain proprietary for internal Hitachi R&D use.
  - "command device" based functionality is not available to the public.
- Ivy presentation material and sample demo output & videos describe Hitachi command device connector functionality so as to
  - Show what ivy was designed to do
  - Serve to illustrate the functionality that other vendors could achieve by developing their own connectors.

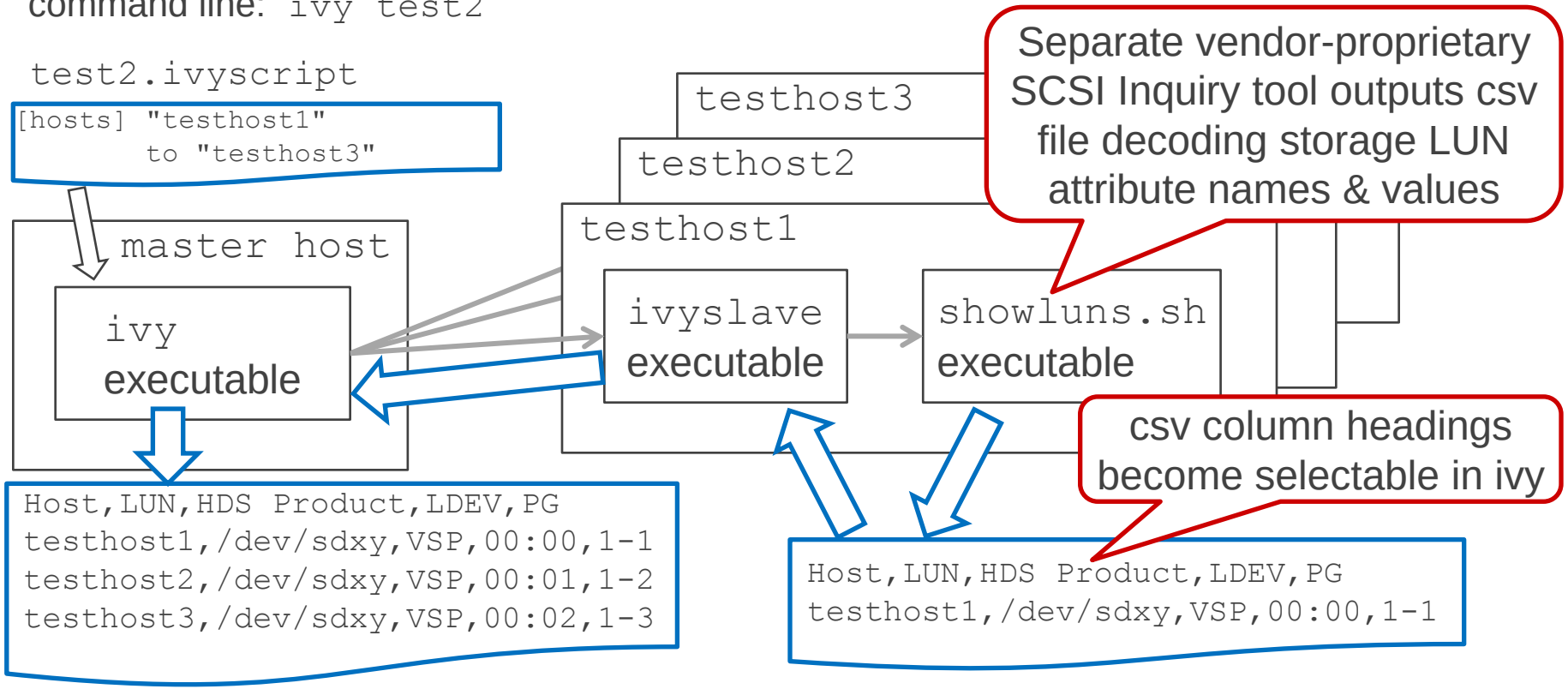
[hosts] statement – configuration discovery

HITACHI
Inspire the Next

command line: `ivy test2`

test2.ivyscript

```
[hosts] "testhost1"
        to "testhost3"
```



[CreateRollup] "LDEV";

HITACHI
Inspire the Next

- A rollup is what mathematicians call a "partition" – a way to divide a collection of things (in this case host-LUN-workload threads) into subgroups.
- There is by default always a rollup called "all" which has one rollup instance, also called "all" which consists of all workloads on all LUNs on all hosts.
- Otherwise, a rollup name must be combination of LUN attribute names
- If you have multiple subsystems under test, to get data rolled up by subsystem by port say

```
[CreateRollup] "serial_number+port";
```

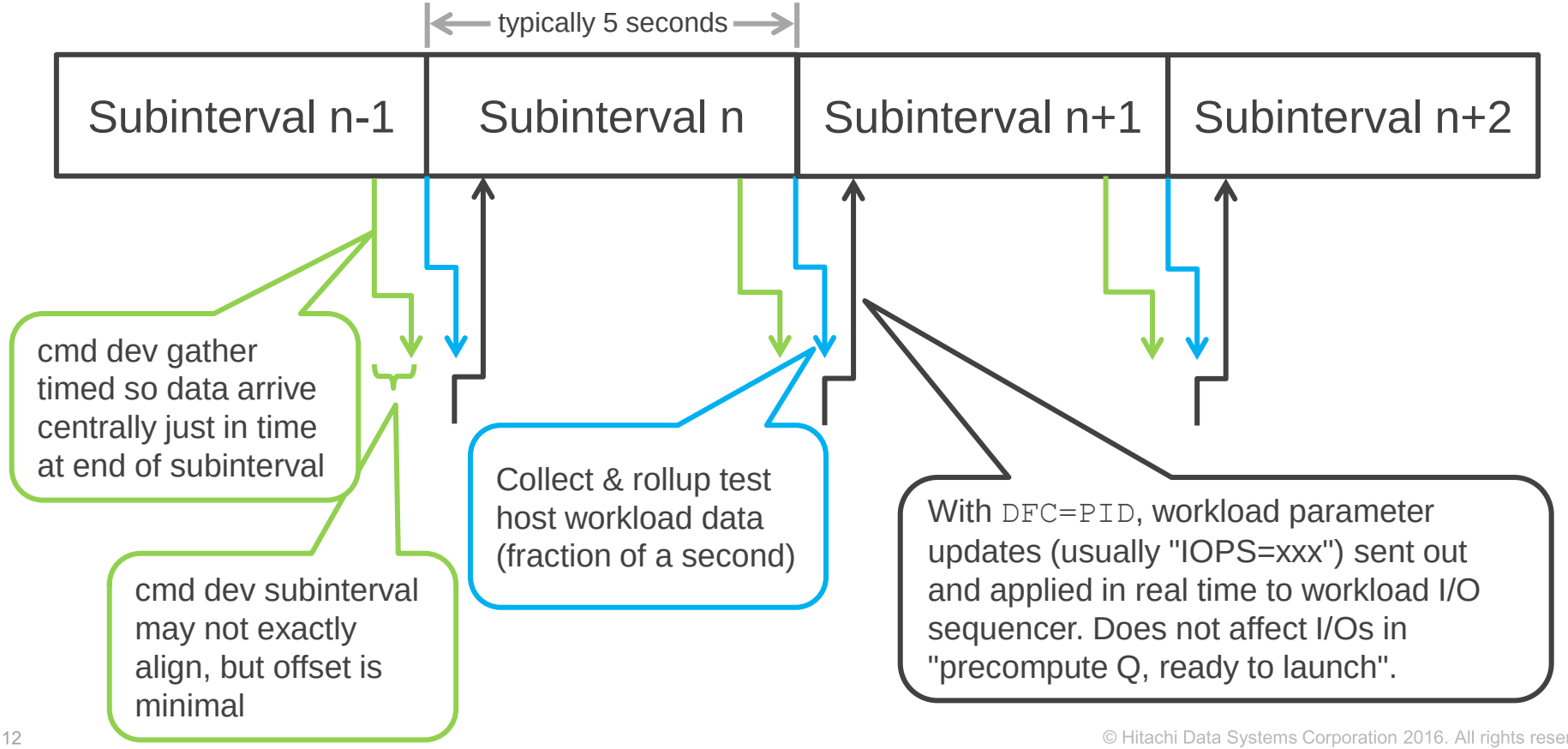
  and get instances like "`410034+1A`".

Rollups are multi-purpose

HITACHI
Inspire the Next

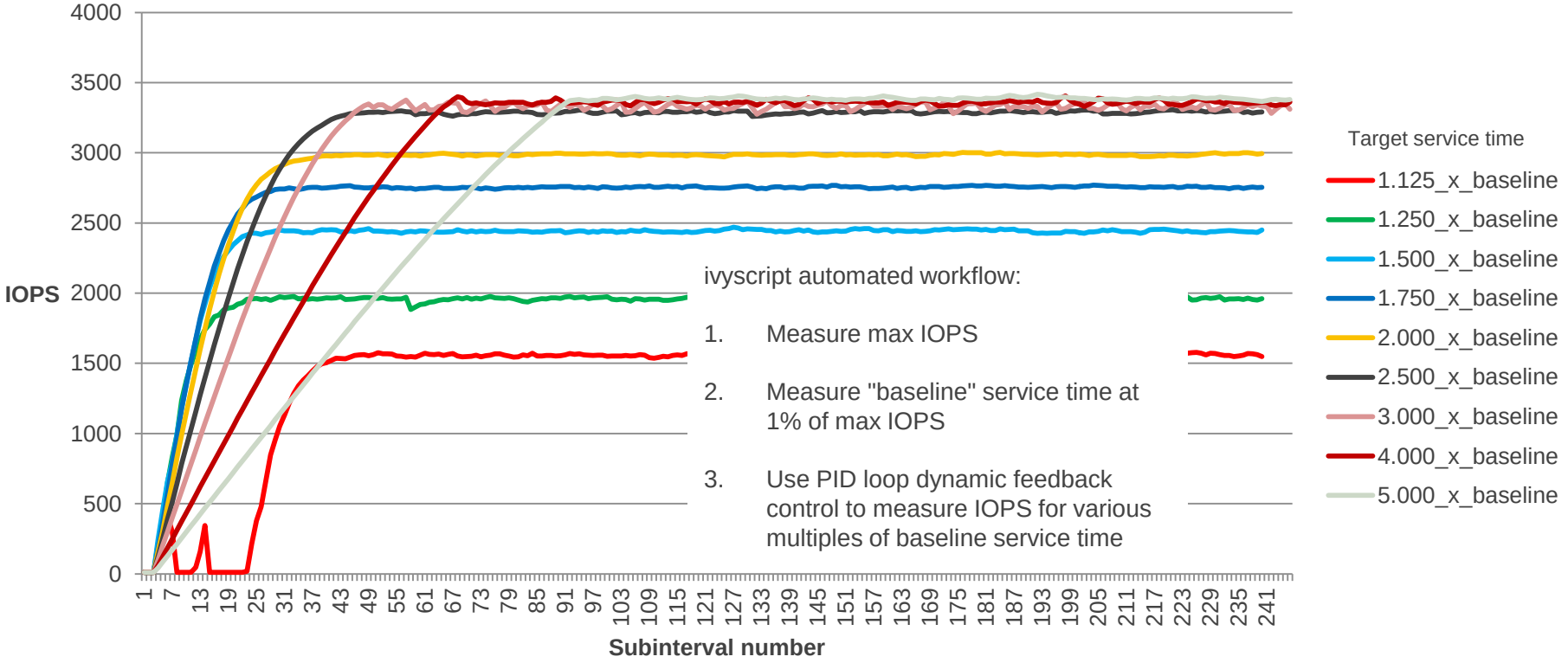
- The default is to print csv files by rollup instance.
  - Say `[nocsv]` to suppress.
- On the `[Go]` statement to run a subinterval sequence, `focus_rollup="PG"`
  - Both `dfc=pid`, the dynamic feedback control feature, as well as `measure=on`, the "seen enough and stop" `[Go]` statement feature operate at the granularity of the rollup instance.
- `[CreateRollup] "port" [quantity] 32;`
  - Will invalidate measurements if we don't get data for exactly 32 subsystem ports.
- `[CreateRollup] "port" [MaxDroop] "20%";`
  - Invalidates if there is a port whose IOPS is more than 20% slower than the fastest port.

ivy dynamic feedback control (DFC)



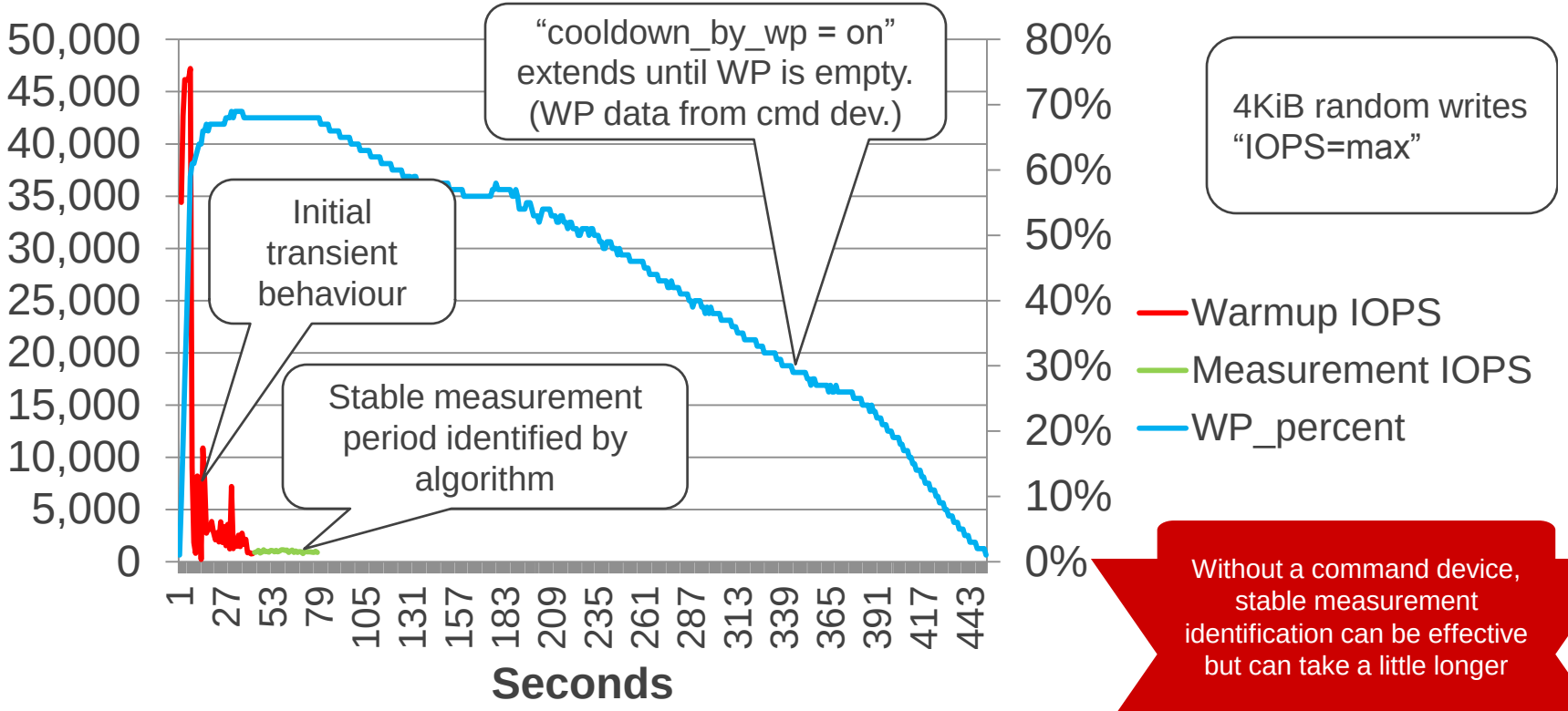
industry standard "PID Loop" controller

HITACHI
Inspire the Next



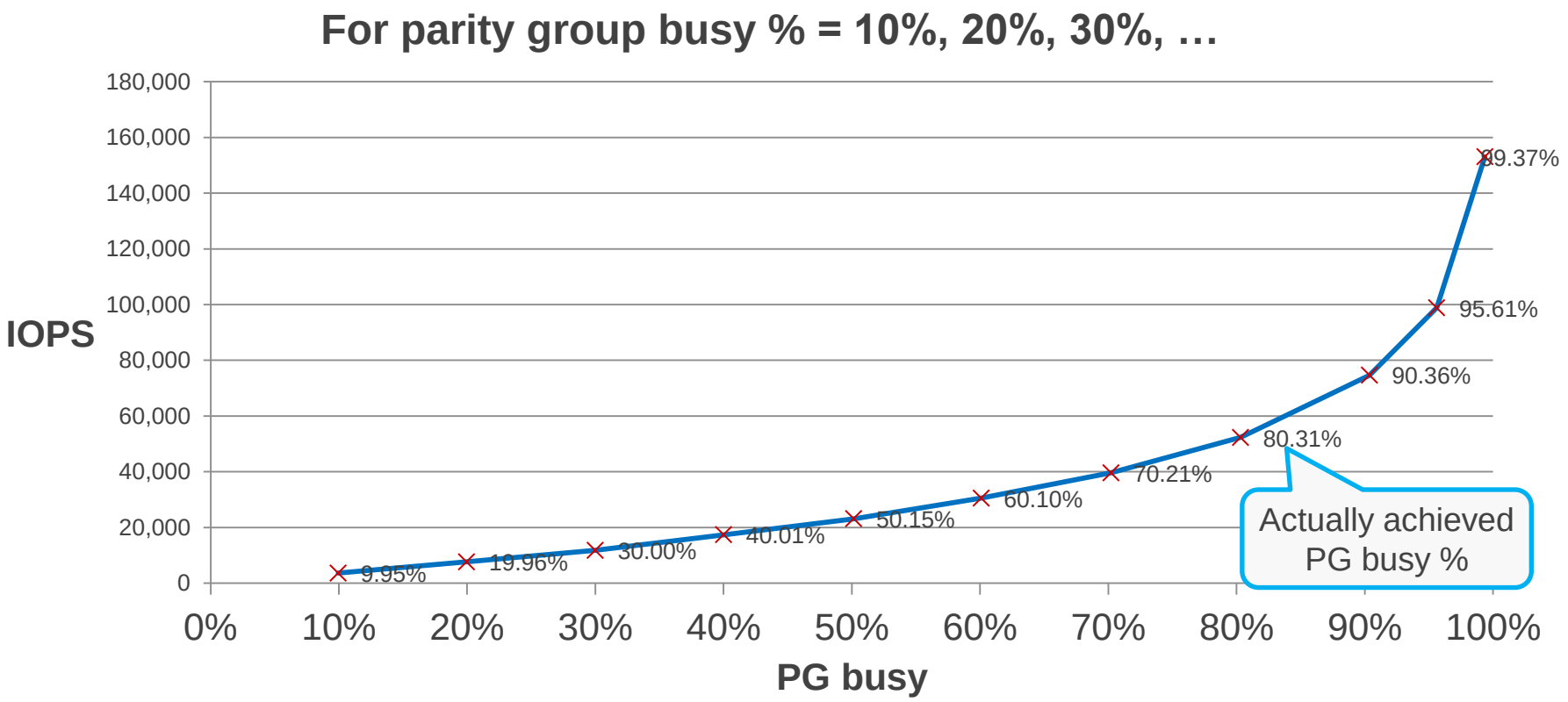
measure=on, cooldown_by_wp=on

HITACHI
Inspire the Next



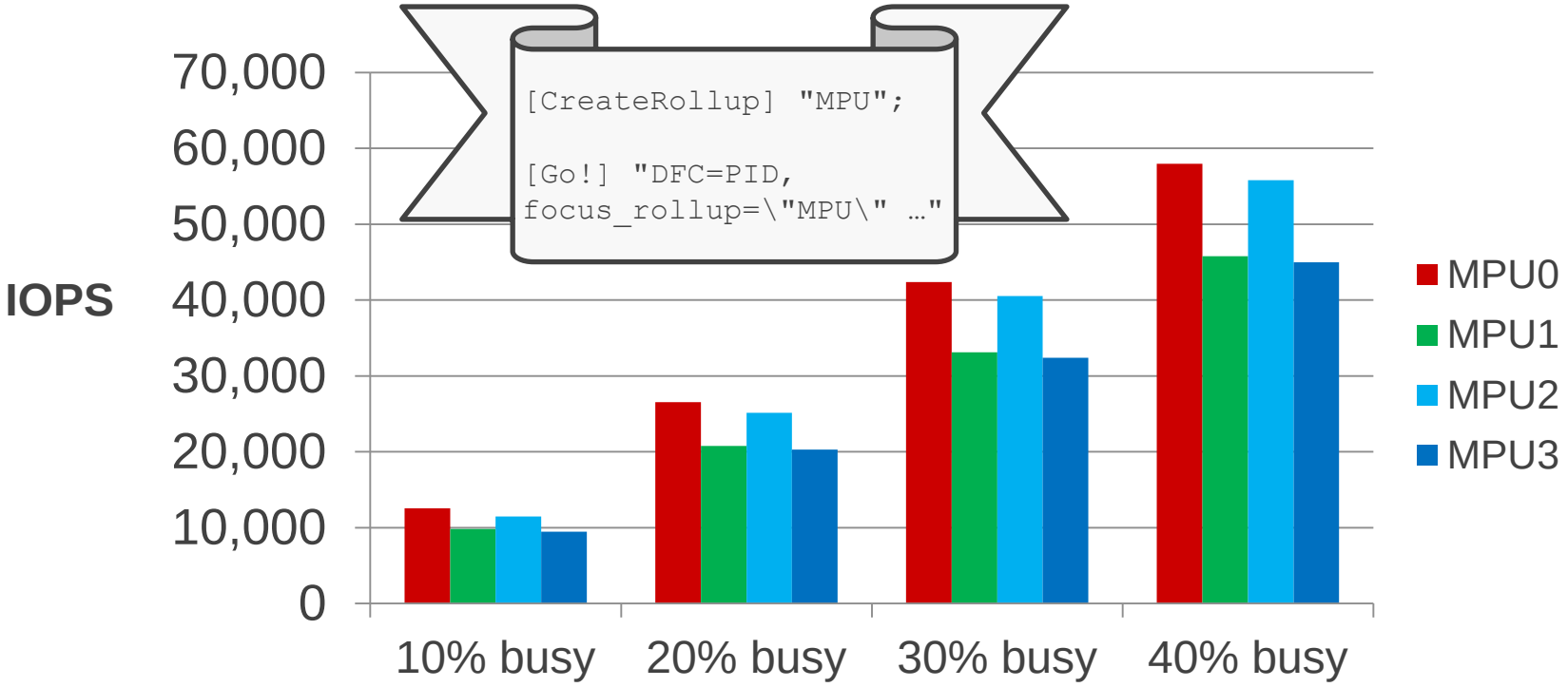
DFC on command device connector data

HITACHI
Inspire the Next

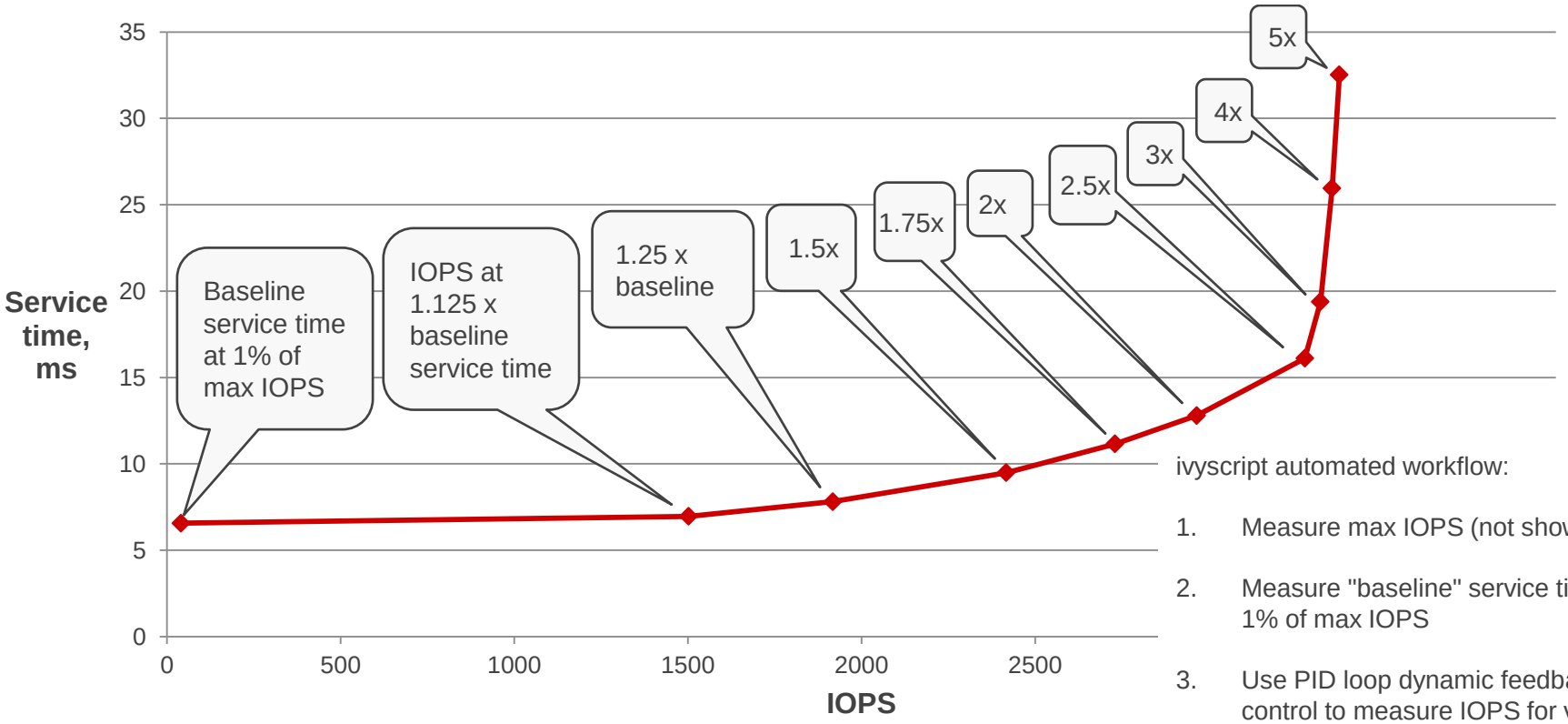


Fine-grained dynamic feedback control

HITACHI
Inspire the Next



DFC on host workload data – non proprietary

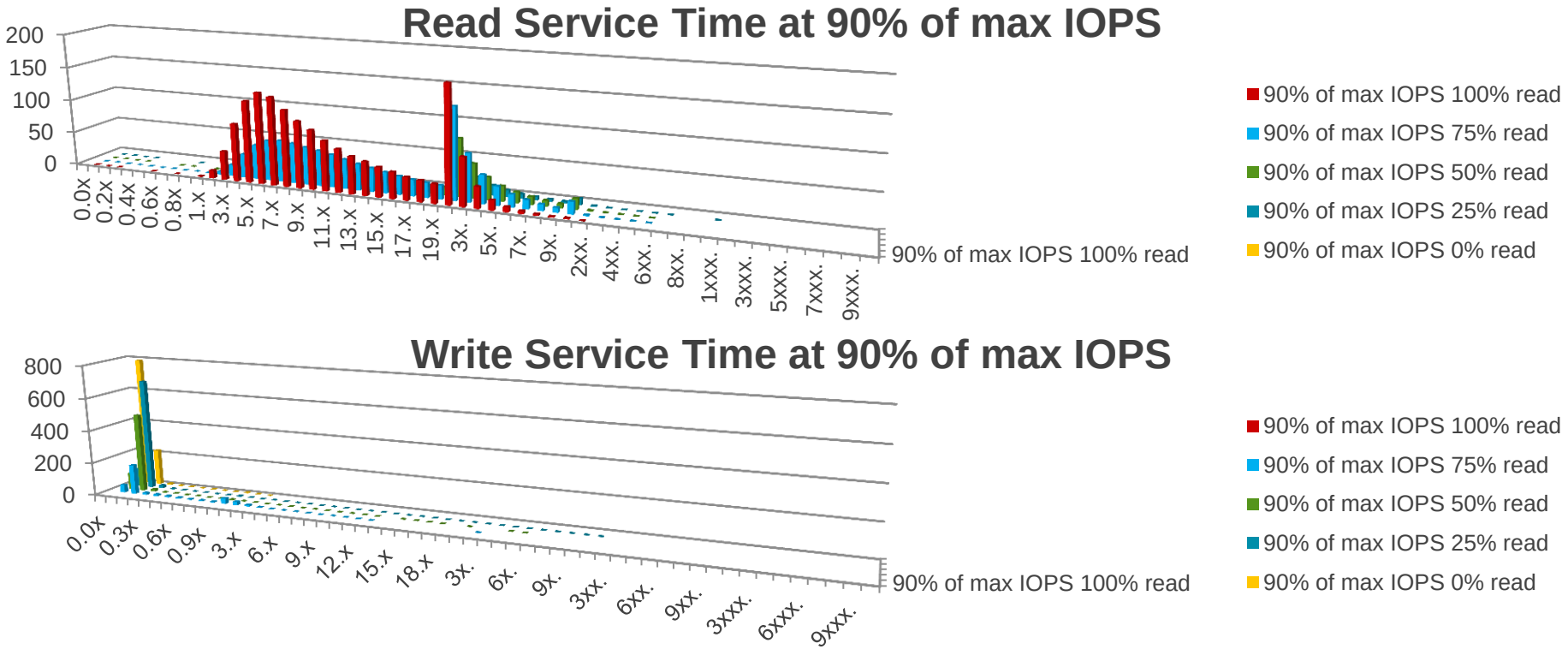


ivyscript automated workflow:

1. Measure max IOPS (not shown)
2. Measure "baseline" service time at 1% of max IOPS
3. Use PID loop dynamic feedback control to measure IOPS for various multiples of baseline service time

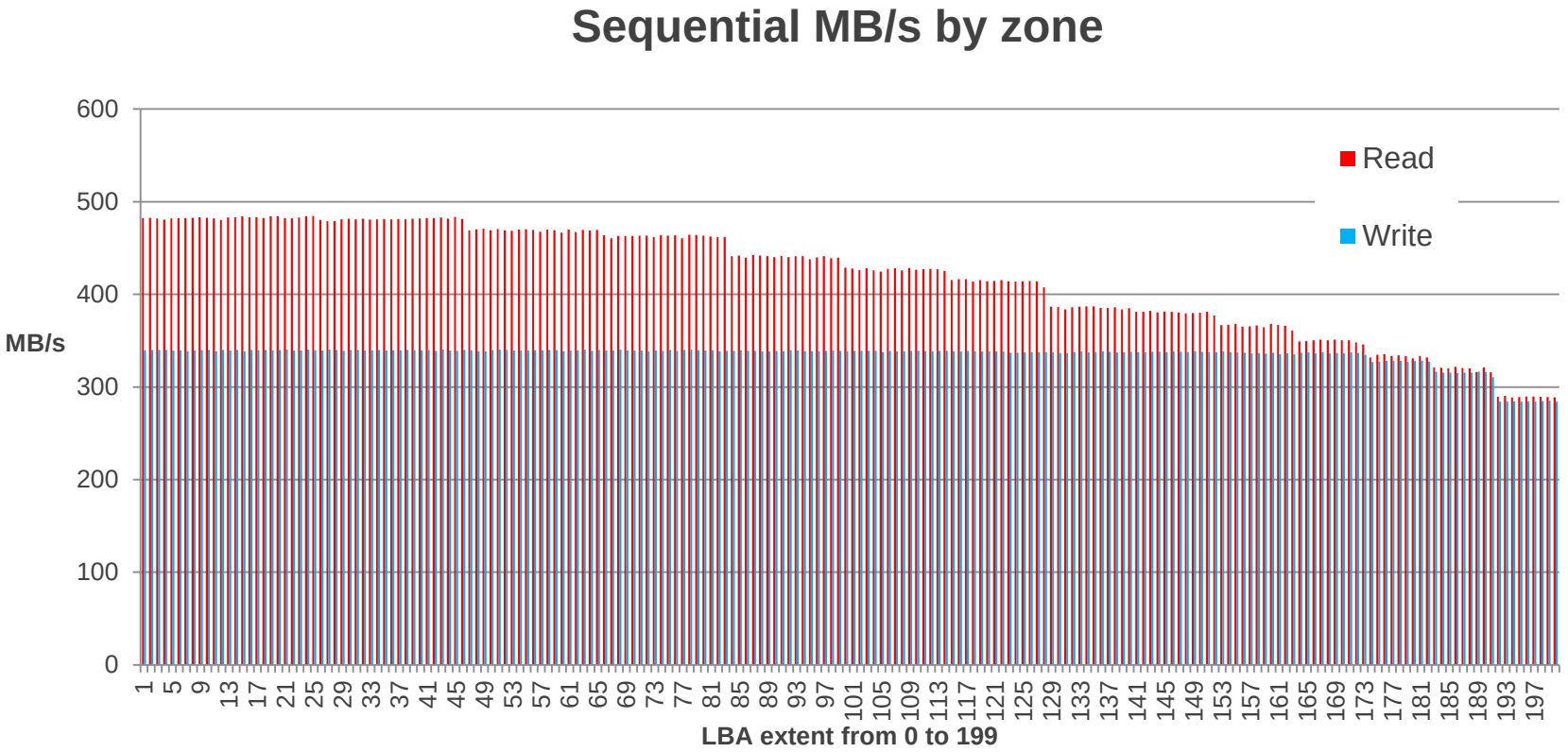
Service time histograms

HITACHI
Inspire the Next



Workloads can be placed within the LUN

HITACHI
Inspire the Next



Ivy development status

HITACHI
Inspire the Next

- May, 2016 initial version.
  - Ready for initial use by Hitachi internally
  - Approved for release as open source release. Now preparing HDS community site for opening for public access by end of May.
  - Demos available for all functionality.
  - No official QA performed (developer testing only), no end user experience yet.
  - Support for VSP Gx00, most, but not all command device data.
  - Support for HUS100 series.

Ivy futures

HITACHI
Inspire the Next

- Need "shakedown" user experience before layering more development
- Dedupe / compression features – in development
- Some ideas for the future:
  - Repackage access to ivy engine via a RESTful API.
    - Build CLI on RESTful API to enable programming ivy in Python or any other language
    - Remove outer ivyscript programming layer wrapper, exposing ivy engine control statements as CLI commands.
  - Develop VSP G1000 support
  - Extend VSP Gx00 support
  - Develop "auto gain control" feature for PID loop
  - Transition to POSIX asynchronous I/O and port to other OS platforms.

Thank You

HITACHI

Inspire the Next