Introduction to "ivy"

Block storage synthetic workload generator with real time dynamic feedback control

https://github.com/Hitachi-Data-Systems/ivy

Hitachi Data Systems - 2016-08-11

Allart Ian Vogelesang ian.vogelesang@hds.com

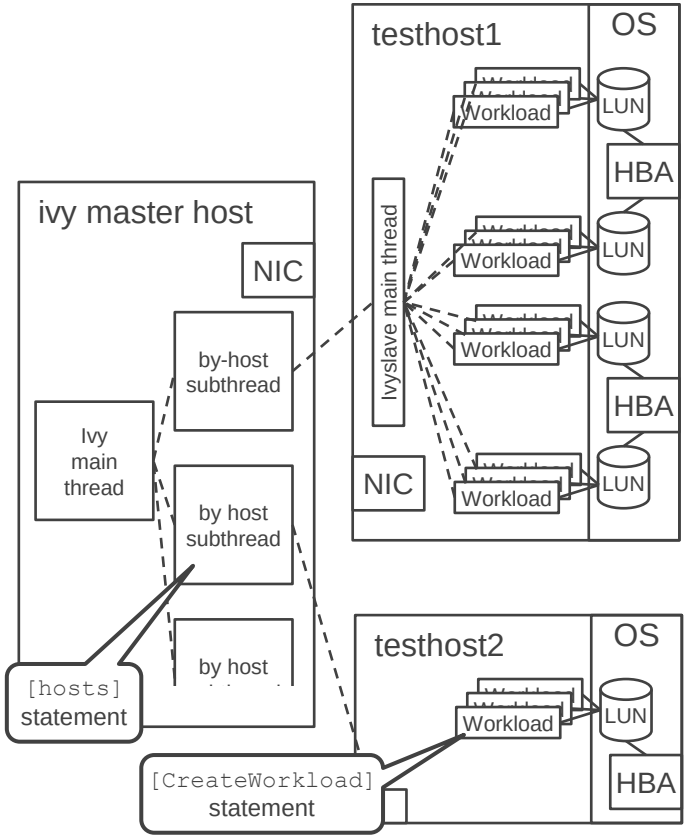
Why ivy? To automate performance testing

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

Scalable block storage synthetic workload generator with real time dynamic feedback control

HITACHI
Inspire the Next



- 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 in one chart

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.
- Super easy test setup – select by LDEV, by Port, by PG, Pool ID
- Can test for minimum time necessary to make a valid measurement to a specified +/- percent accuracy
- Real-time dynamic feedback control (DFC) of workload
  - "measure IOPS at 1 ms service time"
- Records only valid measurement data
  - Can check number of resources reporting (ports, PGs, LDEVs, etc.) and that none running too slowly.
- Workflow automation using ivyscript programming language wrapper around ivy engine.

ivy – achieving vendor independence

HITACHI
Inspire the Next

1. An external vendor proprietary simple SCSI Inquiry-based LUN discovery tool is used.
   - The external tool provides a "discovered LUNs" csv file with a header line
     - ```
       host,LUN name,HDS product,port,LDEV,PG,CLPR, …
       testy1,/dev/sdxy,VSP,1A,00:00,1-1,CLPR0, …
       testy1,/dev/sdyz,VSP,2A,00:01,1-2,CLPR0, …
       ```
   - To support another vendor's architecture or terminology, all you need is the external LUN discovery tool that produces such a csv file.
   - Hitachi LUN discovery tool is open source - https://github.com/Hitachi-Data-Systems/LUN_discovery
2. In ivy, the LUN discovery csv file header line column titles become selectable
   - `[select] "port" is { "1A", "3A", "5A", "7A"}, "CLPR" is "CLPR0"`
       ```

Proprietary extension – not open source

HITACHI
Inspire the Next

- The "command device connector", or `ivy_cmddev` executable.
  - Available only for authorized use in Hitachi labs, with license key mechanism.
  - Transparently / automatically connects to a command device if one is available.
- Retrieves subsystem configuration data
  - Enables selection of test configuration by subsystem configuration attributes such as `drive_type`.
- Retrieves real time subsystem performance data, aligned with test subintervals.
  - Records what is happening inside the subsystem correlated with what the hosts are seeing.
  - Enables real-time dynamic feedback control of host workload based on subsystem data
    - *"measure IOPS at 50% owning MP core % busy", or "measure IOPS at 50% parity group percent busy"*
- Checks that subsystem does not have any failed components.

Other vendors – make your own 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 subsystem configuration attributes.
  - 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.

Demos show command device connector functionality

HITACHI
Inspire the Next

- Why, when only authorized Hitachi lab users have access?
- 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.

[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`".

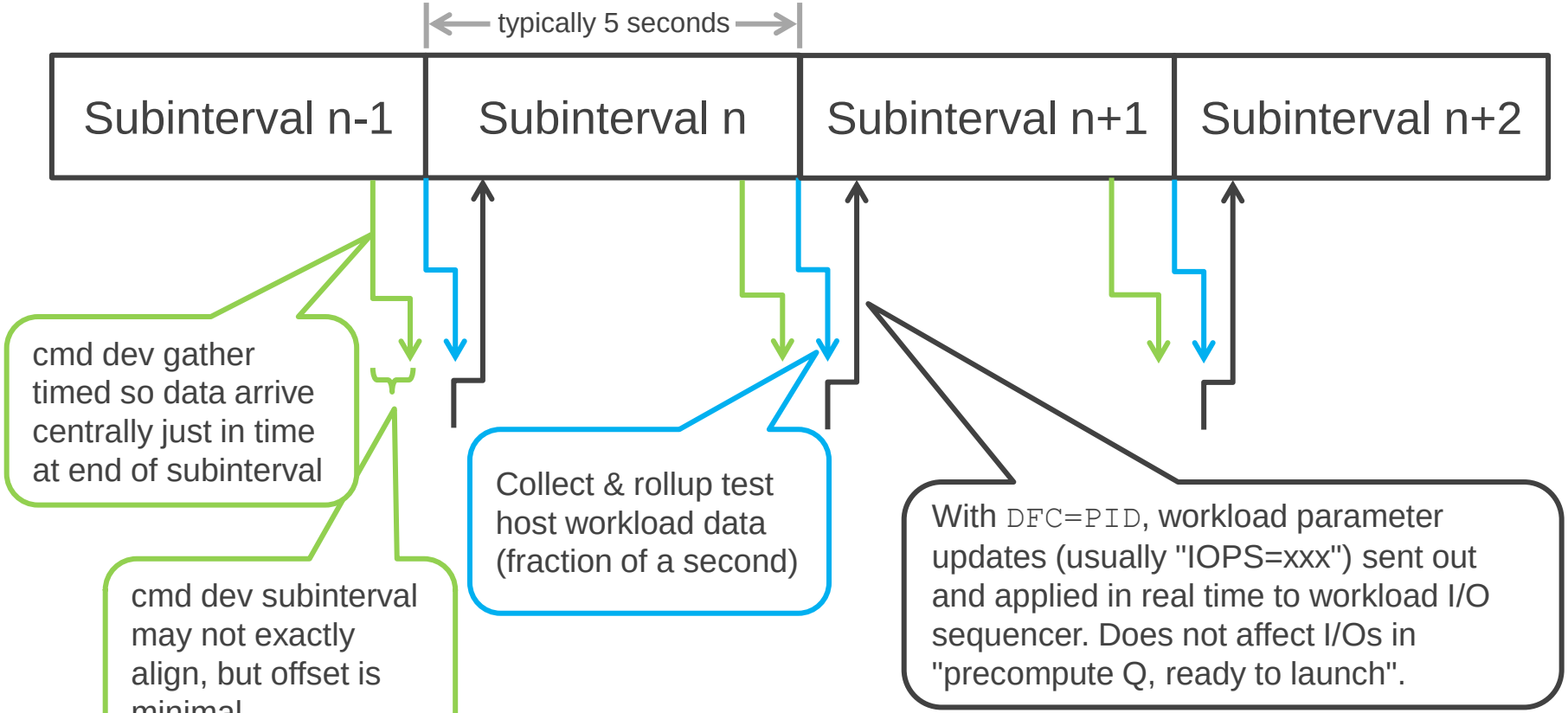
Rollups are multi-purpose

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`, the "seen enough and stop" 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.

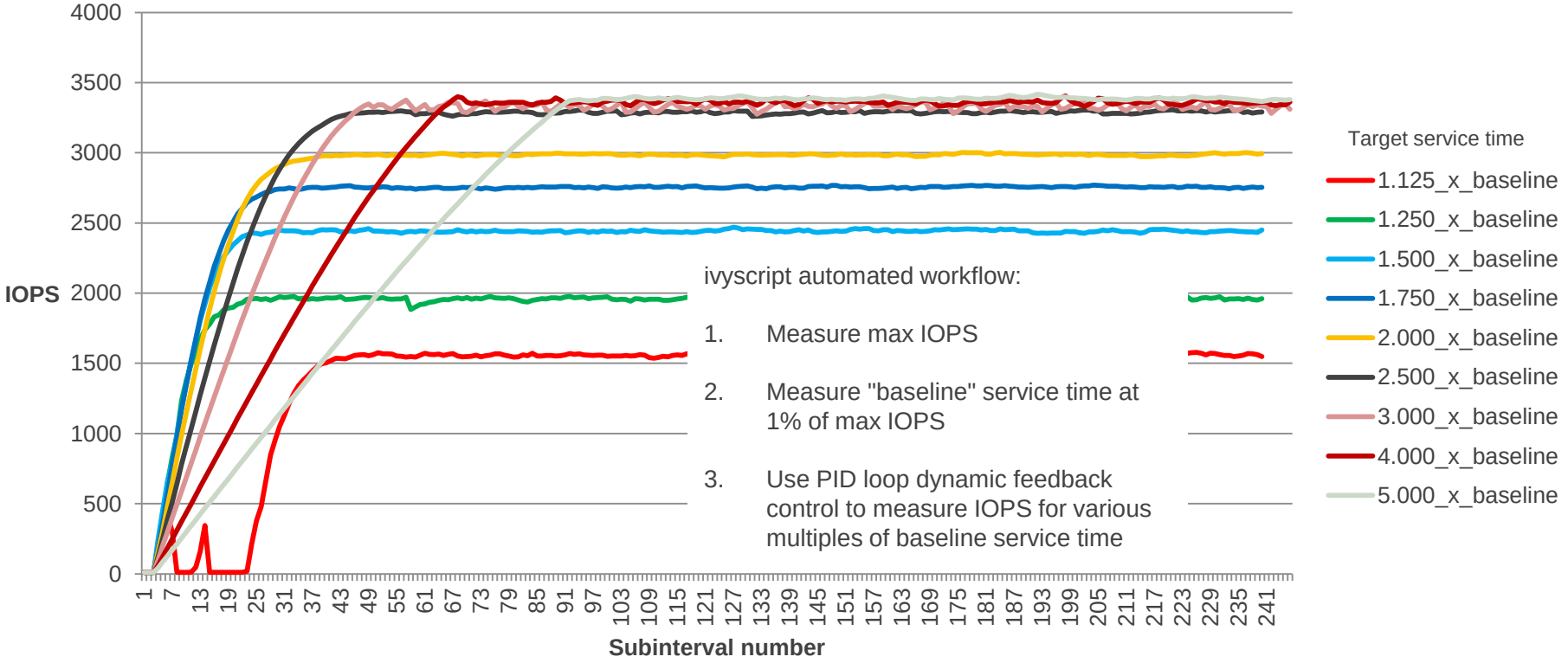
ivy dynamic feedback control (DFC)

HITACHI
Inspire the Next

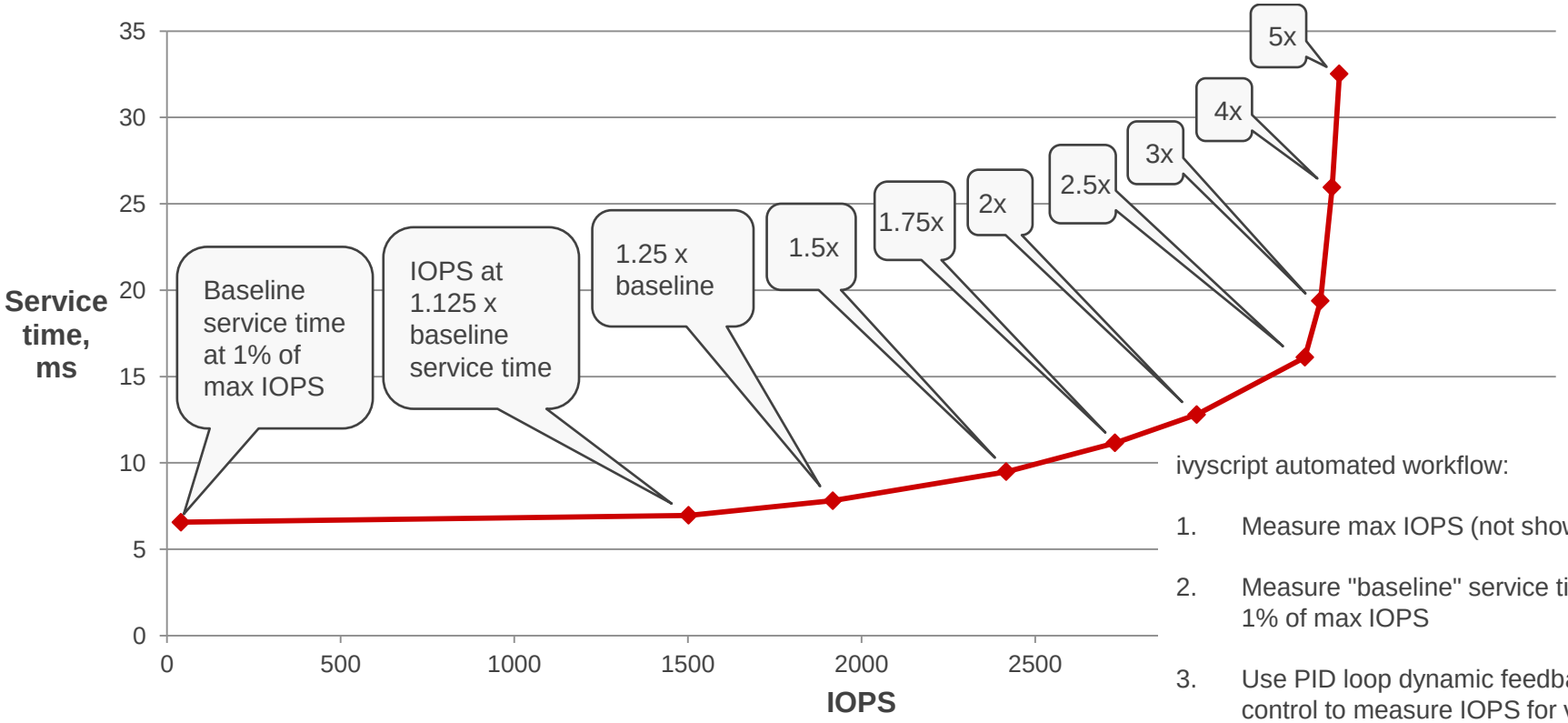


industry standard "PID Loop" controller

HITACHI
Inspire the Next



DFC on host workload data – non proprietary

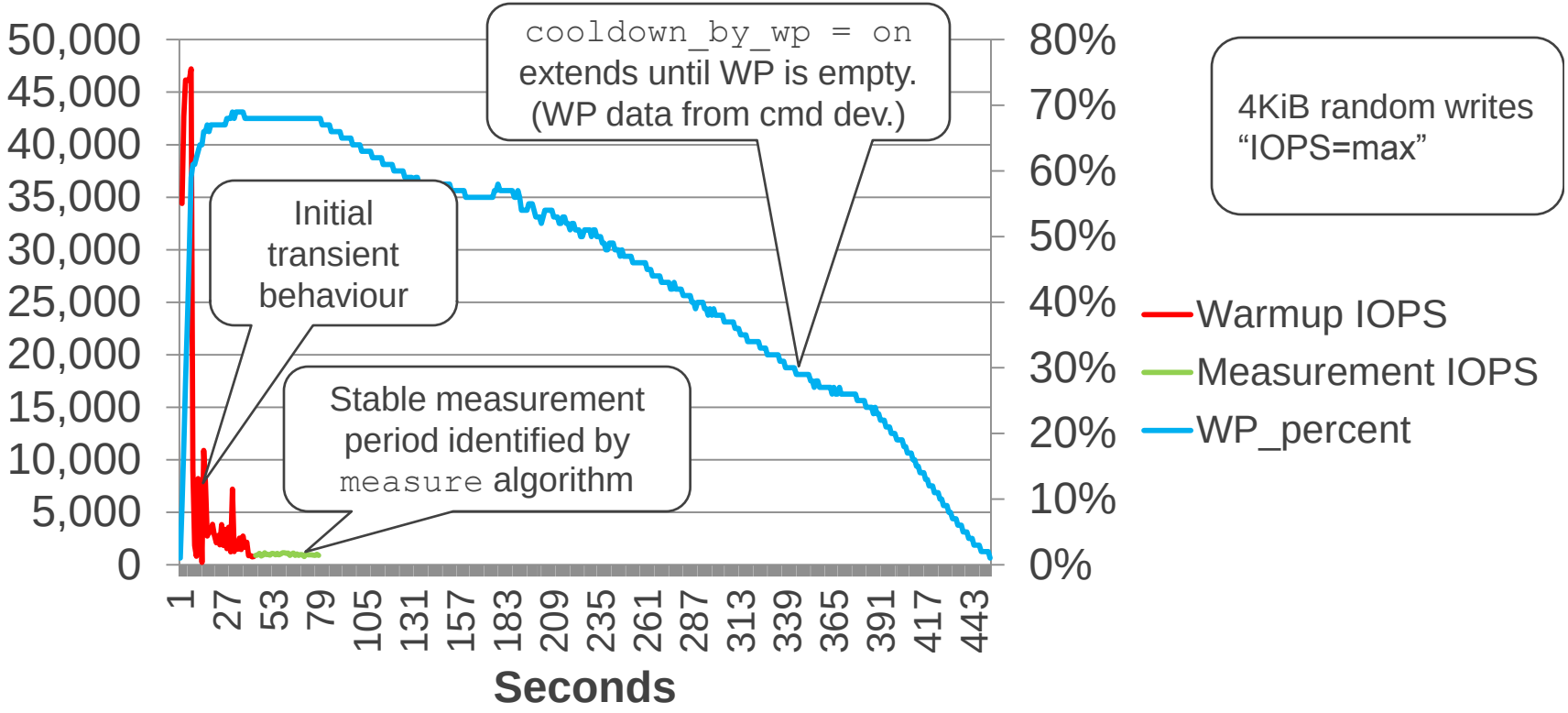


ivyscript automated workflow:

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

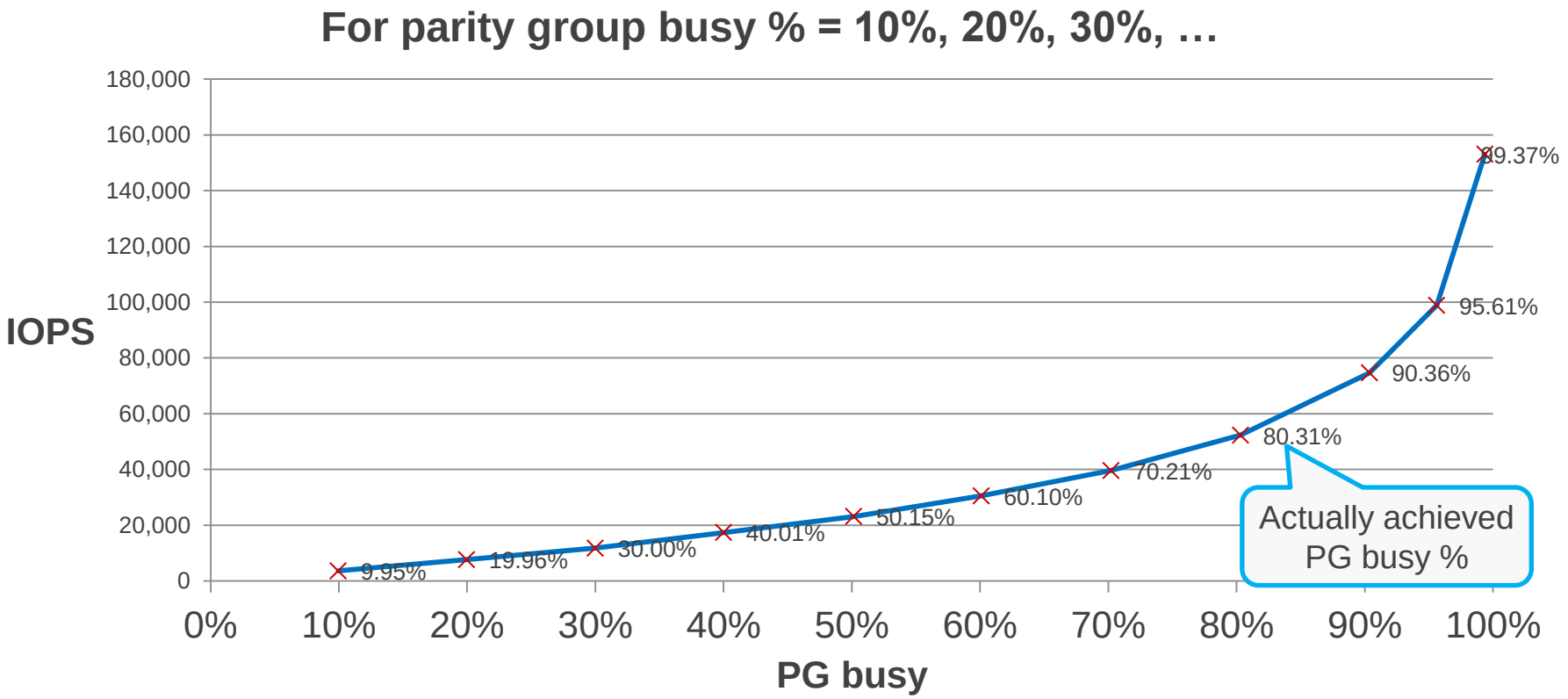
measure=IOPS, cooldown_by_wp=on

HITACHI
Inspire the Next



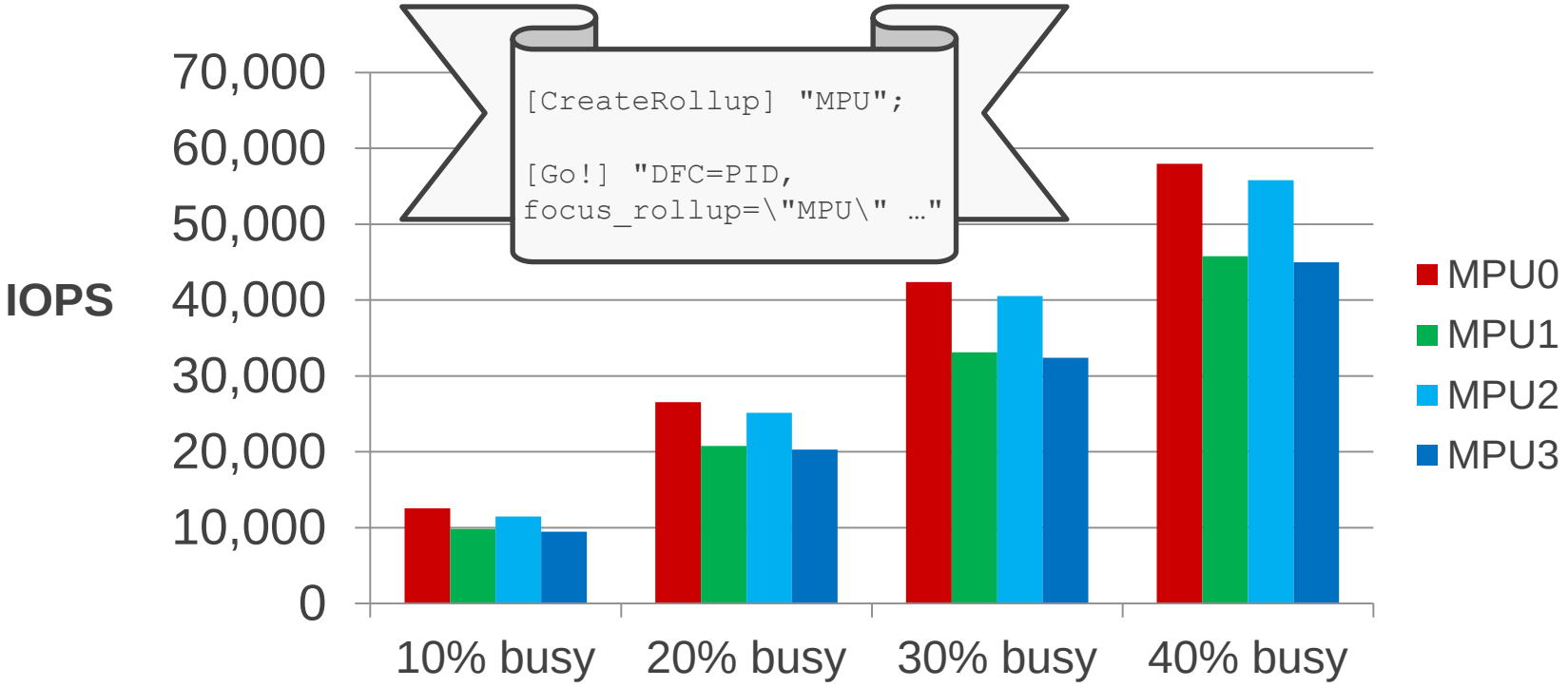
DFC on command device connector data

HITACHI
Inspire the Next



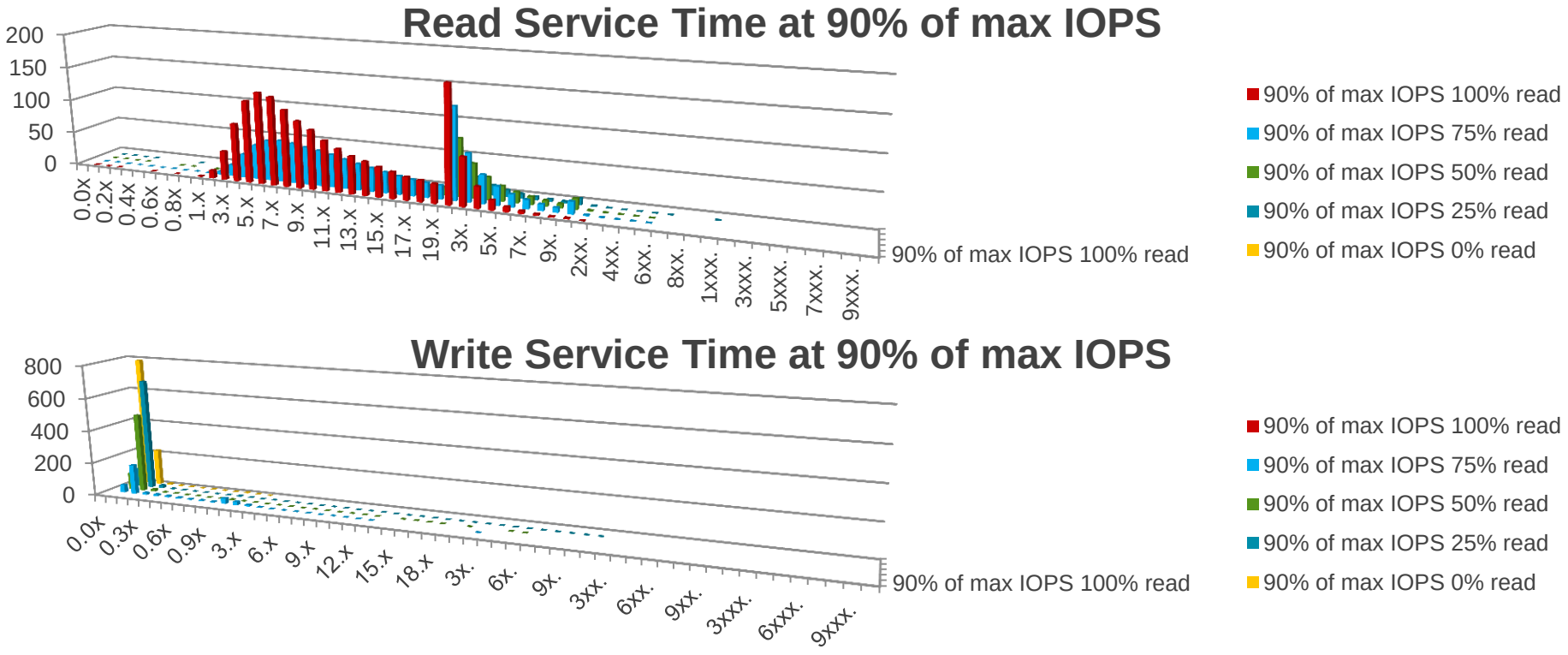
Fine-grained dynamic feedback control

HITACHI
Inspire the Next



Service time histograms

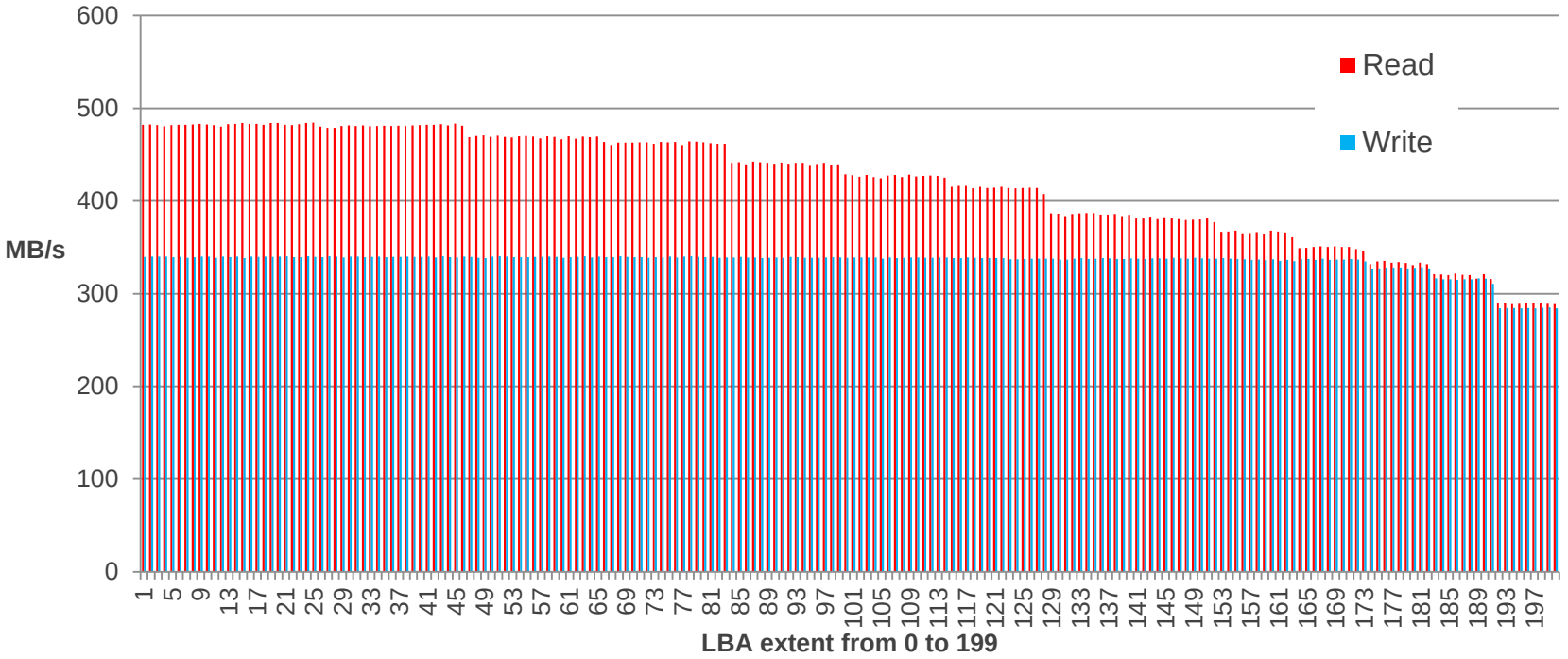
HITACHI
Inspire the Next



Workloads can be placed within the LUN

HITACHI
Inspire the Next

Sequential MB/s by zone



Ivy development status

HITACHI
Inspire the Next

- August, 2016
  - Internal HDS performance team user beta test in progress, including test of dedupe & compression support
  - Released as open source.
  - Demos available for all functionality.
  - Support for HUS VM, VSP Gx00.
  - Support for HUS100 series.

Ivy futures

HITACHI
Inspire the Next

- Some ideas:
  - 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