
# Parallel I/O Architecture Modelling Based on File System Counters

June 23, 2016 | **S. El Sayed**[JSC] M. Bolten[Kas] and D. Pleiter[JSC] |
[JSC] Jülich Supercomputing Centre, Forschungszentrum Jülich

[Kas] Institut für Mathmatik, Universität Kassel

JÜLICH FORSCHUNGSZENTRUM

# CONTENTS

JÜLICH
FORSCHUNGSZENTRUM

# Parallel I/O Architecture Modelling Based on File System Counters

## Part I: Motivation

June 23, 2016 | **S. El Sayed**[JSC] M. Bolten[Kas] and D. Pleiter[JSC]

# Motivation

## Why simulate?

- I/O to compute imbalance
  - Increase in compute performance not met with a matching increase in I/O
- Applications I/O requirement is increasing

**Solution:** Emerging I/O architectures

- Hierarchical storage
- Active storage

**Key Point**

Considering current application I/O behaviour:
**Can emerging I/O architectures improve I/O performance?**

JÜLICH
FORSCHUNGSZENTRUM

# Parallel I/O Architecture Modelling Based on File System Counters

## Part II: Methodology - Model Creation and Validation

June 23, 2016 | **S. El Sayed**[JSC] M. Bolten[Kas] and D. Pleiter[JSC]

# Model Creation

Define

1. Model components
   - High level abstraction vs. detailed components
2. Component behaviour
   - Interaction between components
3. Component parameters
   - Quantify interaction
   - Number of parameters
     (Risk: Parameter sensitivity and over fitting)

**Ex.:** SSD components, behaviour and parameters

Detailed model:

| | | |
|---|---|---|
| Components | : | Internal chips, ... etc. |
| Behaviour | : | Wear leveling, ... etc. |
| Parameters | : | Block size, ... etc. |

High level abstraction:

| | | |
|---|---|---|
| Components | : | Single unit |
| Behaviour | : | Fulfil I/O requests |
| Parameters | : | Bandwidth Read/Write |

# Model Validation

Compare model predictions to test data set

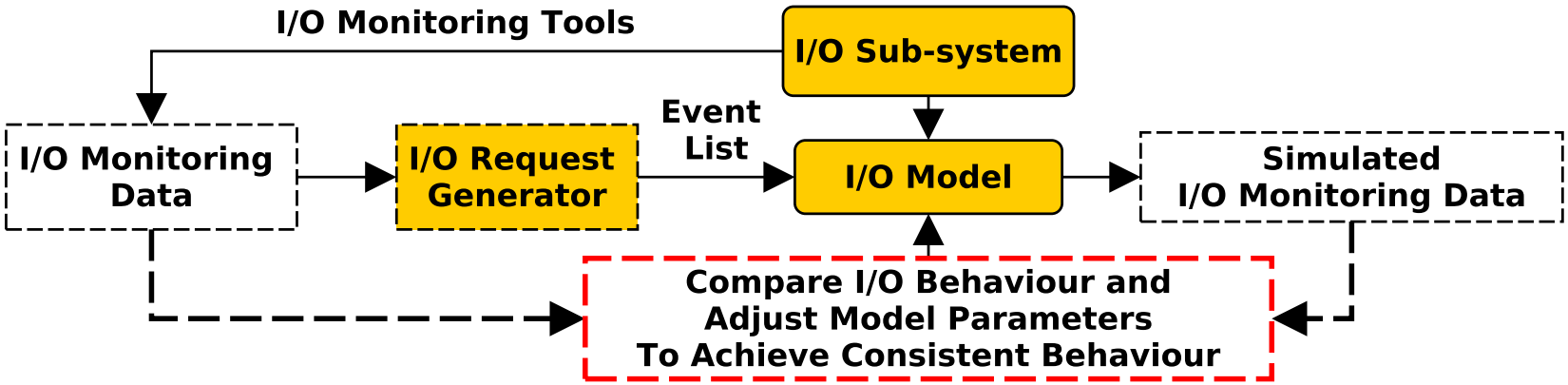
- I/O monitoring
  - Application profiling, Ex. `VAMPIR`
  - Using library profiling, Ex. `DARSHAN`
  - System wide data collection, Ex. `SIOX`
  - Server-side I/O monitoring, Ex. File system I/O counters

## Assumption

The I/O sub-system allows to regularly log performance counters, such as: amount of data read and written, number of read and write operations, etc.

# Model Validation

## Validation Cycle




## I/O Request Generator

For server-side I/O, it translates the I/O monitoring data to an event list that drives the I/O model, where it's behaviour depends on the I/O monitoring data:

- Type
- Resolution in space and time

# Design Space Exploration

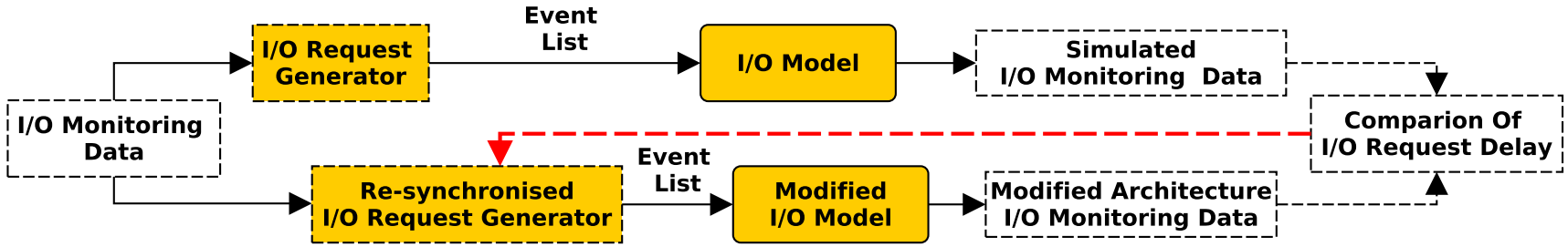
## Main goal

Explore performance impact of (not too large) changes to the architecture

### Changing I/O model architecture

- Adding components
  - Define behaviour
  - Define new parameters
- Limit to architecture modification under inspection

  Ex. Timing constraints



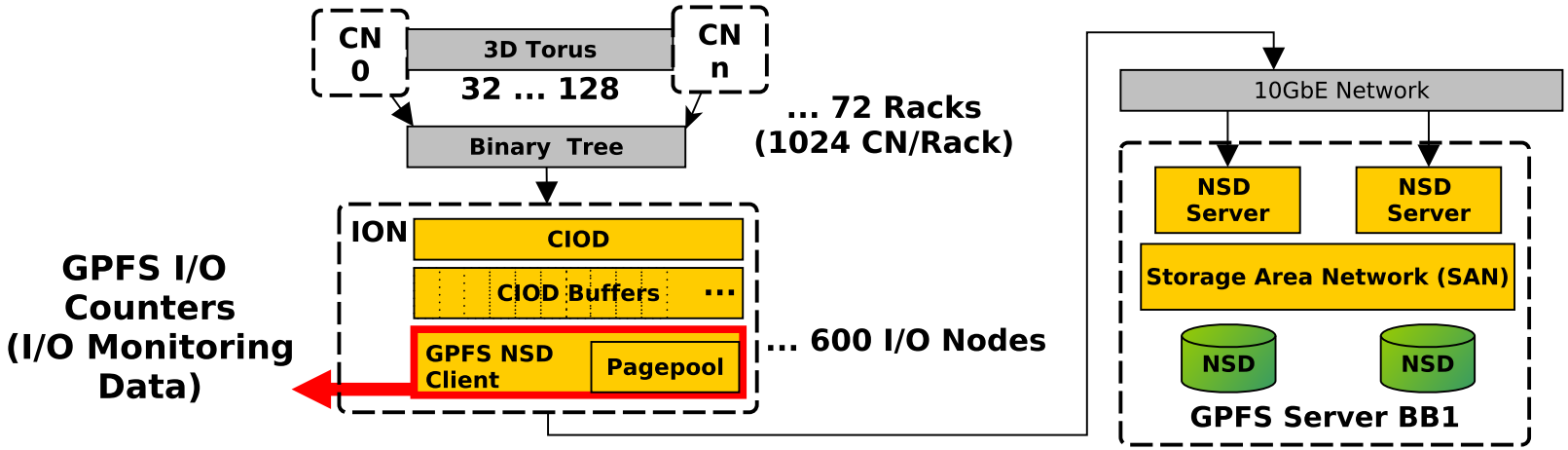

# Parallel I/O Architecture Modelling Based on File System Counters

## Part III: Modelling Blue Gene/P I/O Sub-system (JUGENE)

# Blue Gene/P I/O Sub-system

## Definition

I/O sub-system includes the compute system internal path towards the storage system.




CIOD Control and I/O Daemon
CN Compute Node
ION I/O Node

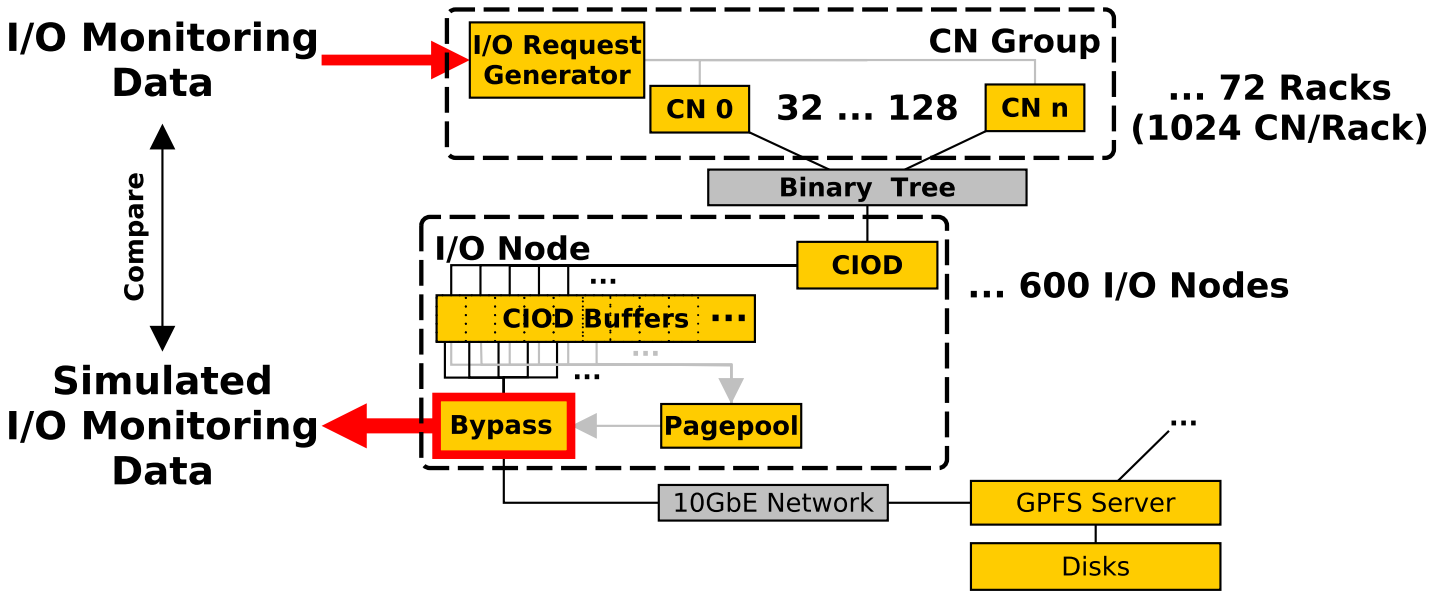
# GPFS I/O Counters

## I/O Monitoring

- Uses the `mmpmon` command from GPFS
- Logs 6 performance counters:
  - Bytes read/written
  - Number of read/write requests
  - Open/close commands
- Temporal: Logged periodically every 120sec
- Spatial: Logged on each ION
- Logged for an extended period of time
  - JUGENE approximately 19 months
  - Continuing on newer systems (JUQUEEN and JURECA)

# Blue Gene/P I/O Sub-system Model

## Components

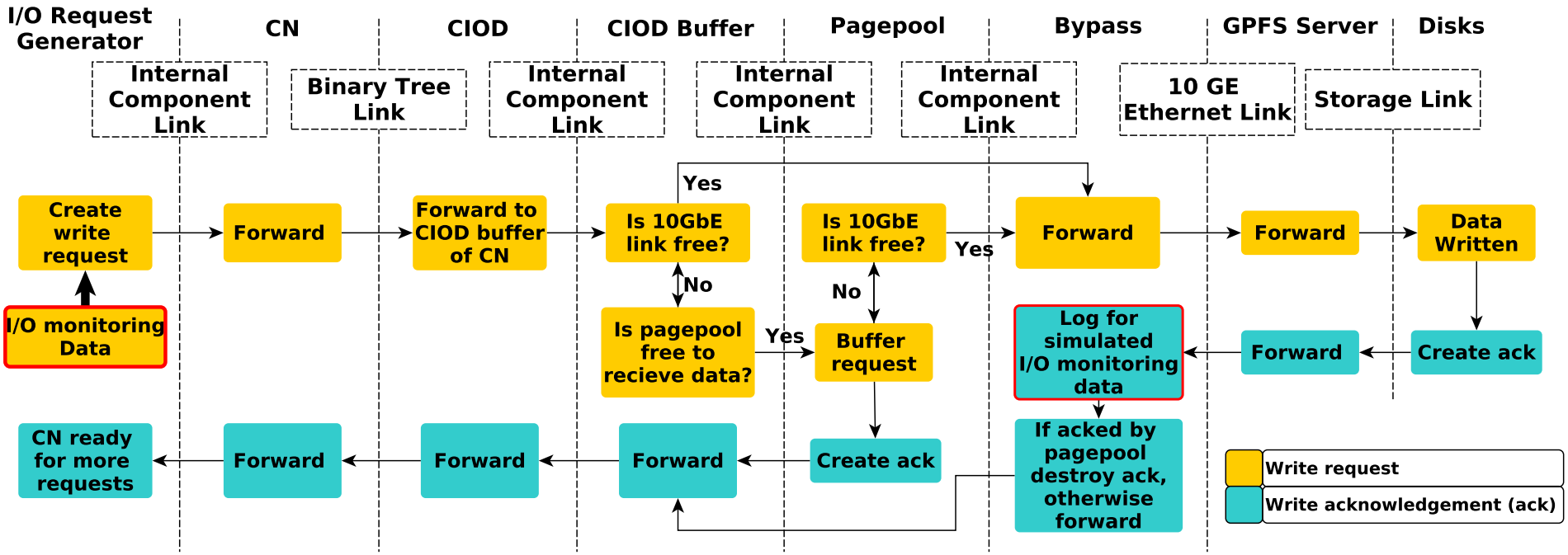



### Data links

- **Internal component link**: zero delay
- **Binary tree link**: CN binary tree that ends at the ION
- **10GE Ethernet link**: Connects the ION to the GPFS server
- **Storage link**: Models the aggregate storage bandwidth

# Blue Gene/P I/O Sub-system Model

## Behaviour




## Flow chart

Define behaviour and interactions of components

# Blue Gene/P I/O Sub-system Model

## Implementation

Simulation framework: OMNET++

- Discrete event simulation framework
- Intended for network simulation
- Modular
- Offers various state capturing and visualizing tools
- C++

# Blue Gene/P I/O Sub-system Model

## Parameters

| Parameter | Initial value |
|---|---|
| Binary tree link bandwidth | 850 MByte/s |
| 10 GE link bandwidth | 1.25 GByte/s |
| Storage link bandwidth | 66 GByte/s |
| CIOD buffer size | 4 MiByte |
| Number of CIOD buffers | 32/128 (CN group size) |
| Page-pool buffer size | 1024 MiByte |

## Validation

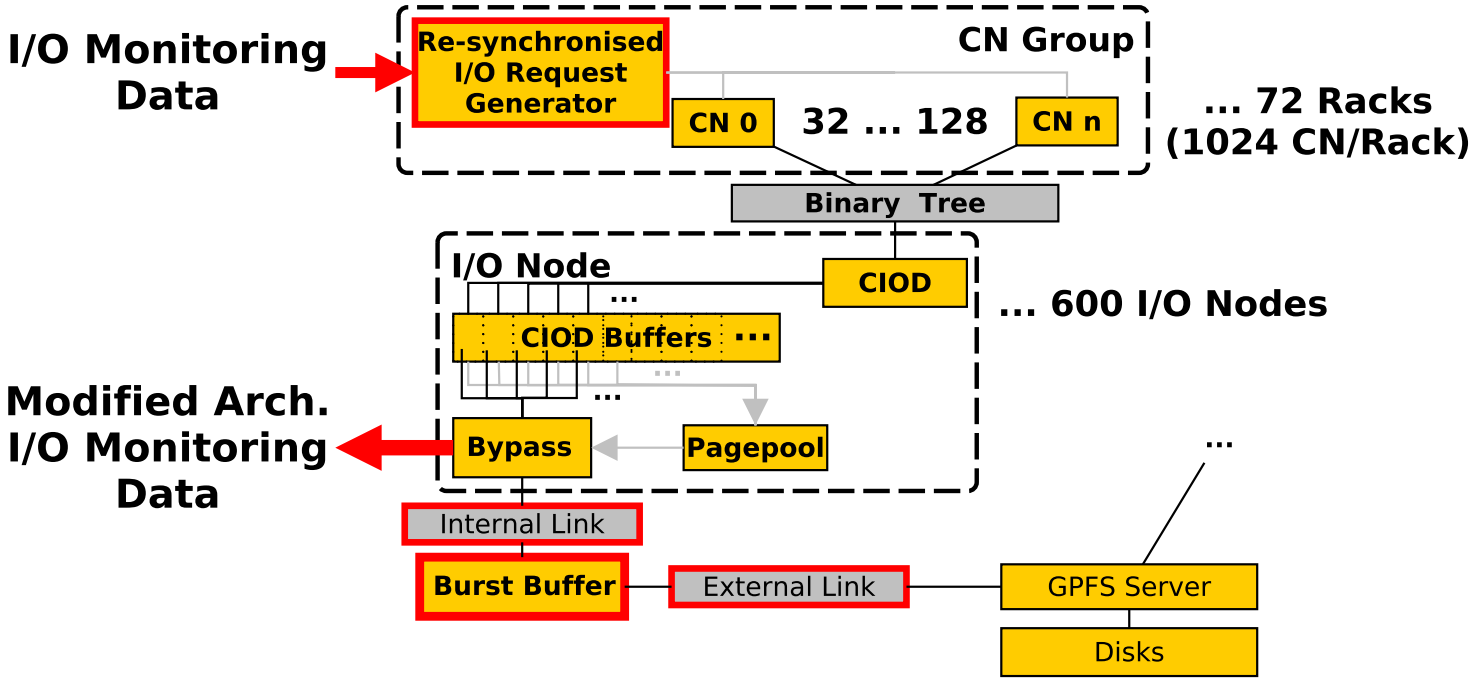
Using initial values for 24h simulated time, the model had an average mismatch of 2% for write requests and less than 1% for read/write bytes and read requests.


# Parallel I/O Architecture Modelling Based on File System Counters

## Part IV: Design Space Exploration

June 23, 2016 | **S. El Sayed**[JSC] M. Bolten[Kas] and D. Pleiter[JSC]

# Burst Buffer I/O Sub-system Model




## New Parameters:

- Internal link bandwidth
- **Burst buffer size**
- **External link bandwidth**

# Burst Buffer Exploration

## Investigation

Reduce cost by decreasing external link bandwidth, while using burst buffers to improve possible performance degradation.

| Case | Burst buffer size | External link bandwidth |
|---|---|---|
| (A) | 16 GiByte | 500 MByte/s = 4x1 GE |
| (B) | 64 GiByte | 125 MByte/s = 1 GE |

Interest:

- Change in I/O time (I/O $\delta t_{bb}/\delta t$)
- Change in job time (Job $\Delta t_{bb}/\Delta t$)

$\delta t$ original I/O time

$\Delta t$ job time

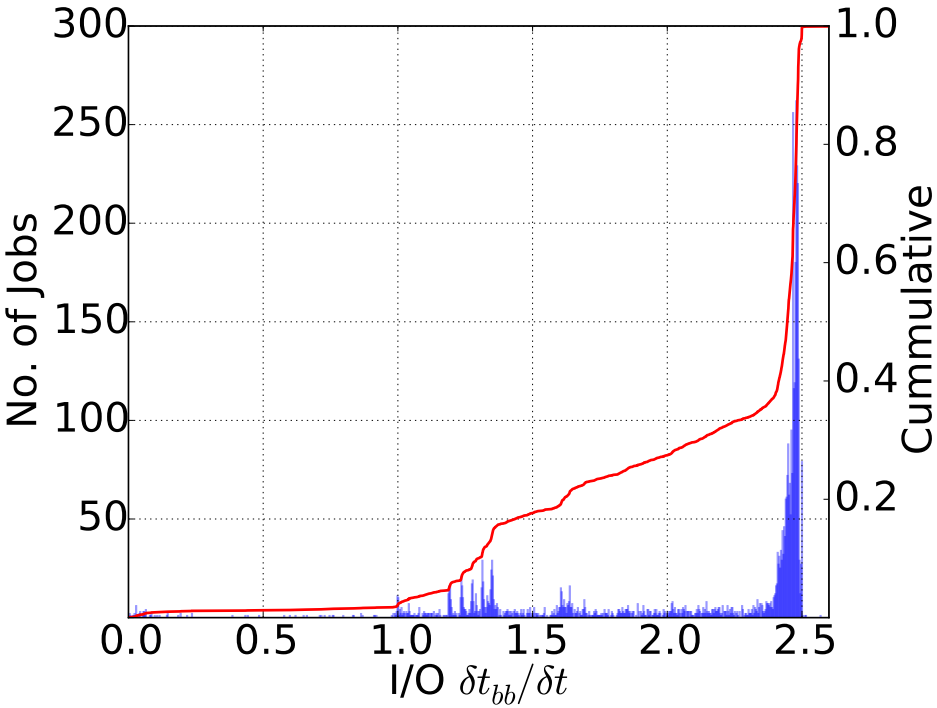
$\delta t_{bb}$ Burst buffer I/O time

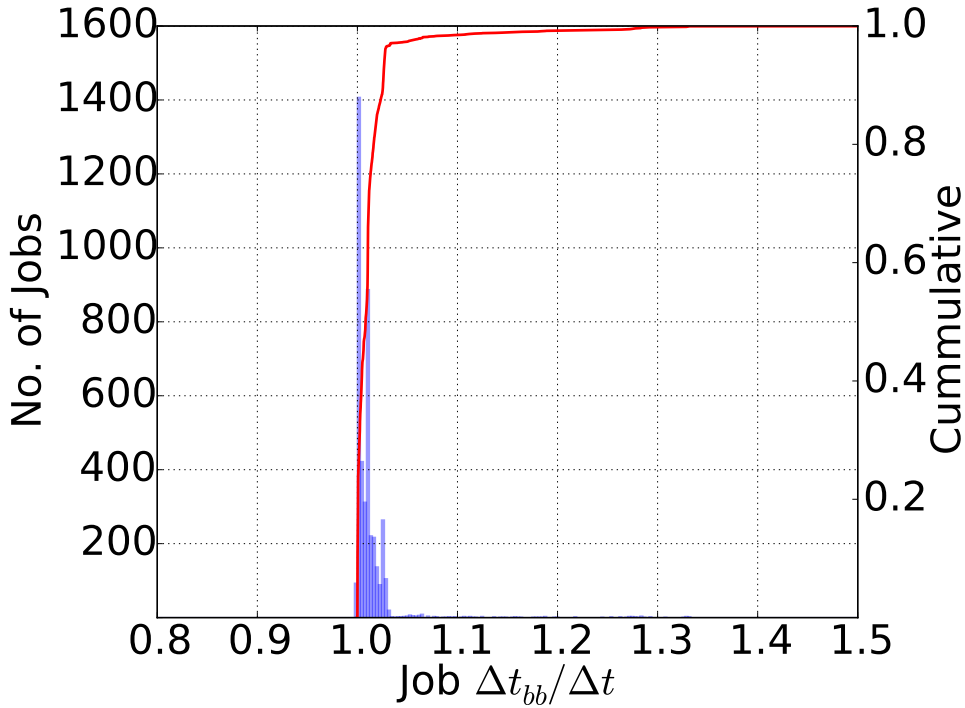
$\Delta t_{bb}$ Burst buffer job time,

# Burst Buffer Exploration

## Case (A)

| Burst buffer size | External link bandwidth |
|---|---|
| 16 GiByte | 500 MByte/s = 4x1 GE |

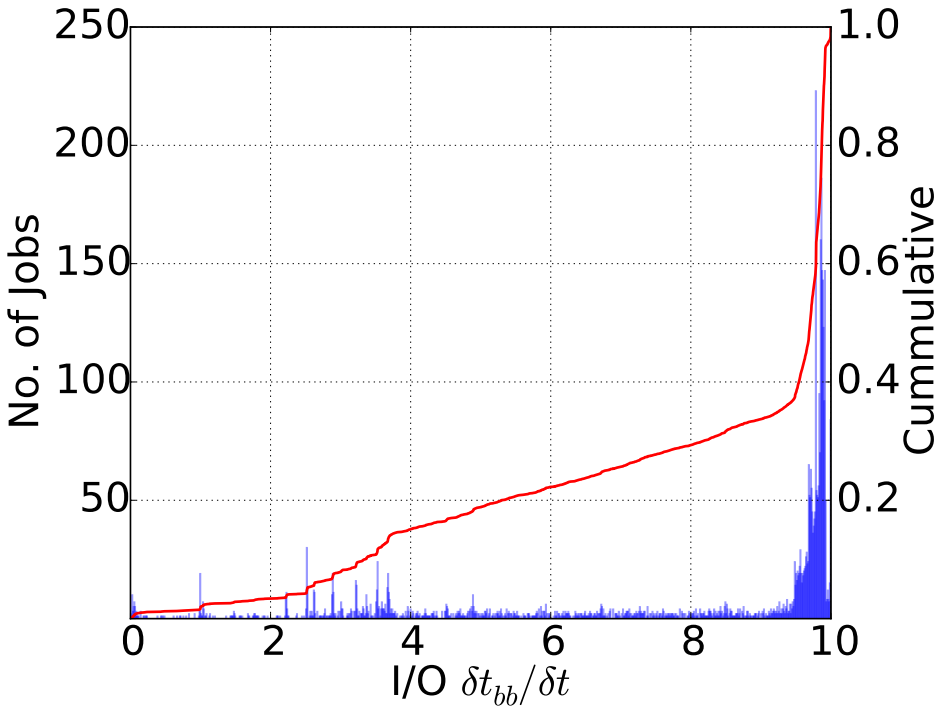


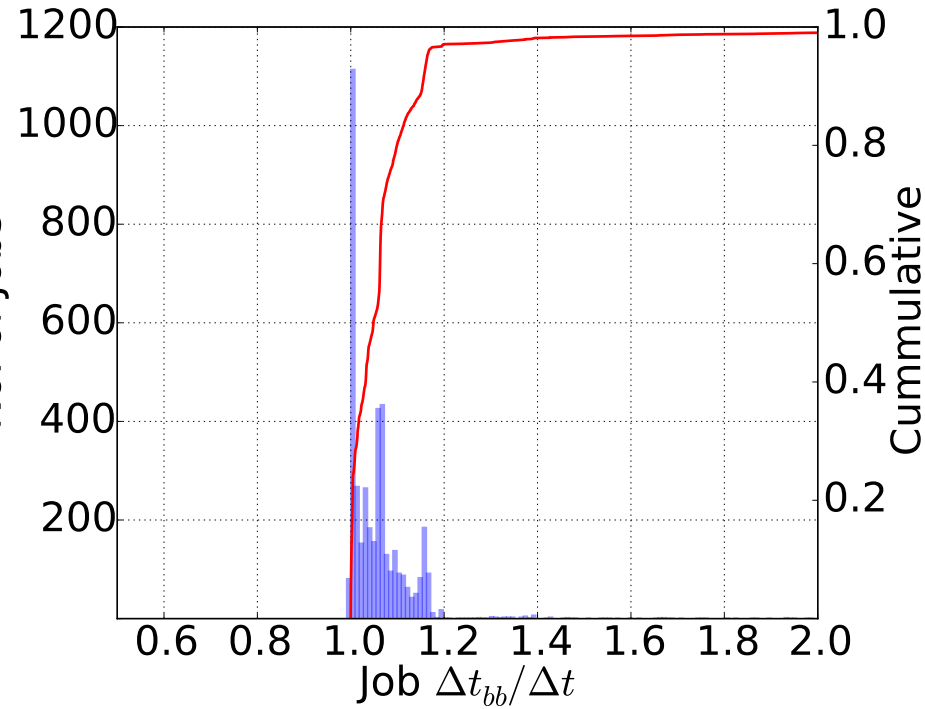



# Burst Buffer Exploration

## Case (B)

| Burst buffer size | External link bandwidth |
|---|---|
| 64 GiByte | 125 MByte/s = 1 GE |



# Burst Buffer Exploration

## Results

- CASE (A)
  - I/O time increased by up to 2.5 times
  - Job time for 90% of jobs increased by less than 5%
- CASE (B)
  - I/O time increased by up to 10 times
  - Job time for 90% of jobs increased by less than 10%
- Performance degradation can be acceptable
  - Given a high system cost reduction
  - Future systems where an increase in external link bandwidth is no longer possible

**Conclusion**

Burst buffers are a good architecture choice and merit more study given applications' I/O behaviour.

JÜLICH
FORSCHUNGSZENTRUM

# Parallel I/O Architecture Modelling Based on File System Counters

## Part V: Summary

June 23, 2016 | **S. El Sayed**[JSC] M. Bolten[Kas] and D. Pleiter[JSC]

# Summary

- Emerging I/O architecture require investigation based on current application I/O behaviour
- Modelling and simulation based on I/O monitoring can offer first look at benefits
- Presented here:
  - Methodology for modelling and simulation
  - Ex: Modelling Blue Gene/P I/O sub-system

JÜLICH
FORSCHUNGSZENTRUM

# Questions?

Thanks