

# Performance Analysis Report: MPI Latency and Bandwidth Regression Tests

Submitted by :

Paul Timileyin Aromolaran – [paul.aromolaran.001@student.uni.lu](mailto:paul.aromolaran.001@student.uni.lu)

Jabin Tasnim Khan Urmey – [jabin.urmy.001@student.uni.lu](mailto:jabin.urmy.001@student.uni.lu)

Ngu Frerashyno Ndah – [frerashyno.ngu.001@student.uni.lu](mailto:frerashyno.ngu.001@student.uni.lu)



```

1 def test_mpi_latency (self):
2     """Test MPI latency on the cluster"""
3     # Run the test
4     mpi_latency = self.run_mpi_latency_test()
5     # Check the results
6     self.assertEqual(mpi_latency, 0.0)
7
8 def test_mpi_bandwidth (self):
9     """Test MPI bandwidth on the cluster"""
10    # Run the test
11    mpi_bandwidth = self.run_mpi_bandwidth_test()
12    # Check the results
13    self.assertEqual(mpi_bandwidth, 0.0)
14
15 def test_mpi_scalability (self):
16    """Test MPI scalability on the cluster"""
17    # Run the test
18    mpi_scalability = self.run_mpi_scalability_test()
19    # Check the results
20    self.assertEqual(mpi_scalability, 0.0)
21
22 def test_mpi_efficiency (self):
23    """Test MPI efficiency on the cluster"""
24    # Run the test
25    mpi_efficiency = self.run_mpi_efficiency_test()
26    # Check the results
27    self.assertEqual(mpi_efficiency, 0.0)
28
29 def test_mpi_stability (self):
30    """Test MPI stability on the cluster"""
31    # Run the test
32    mpi_stability = self.run_mpi_stability_test()
33    # Check the results
34    self.assertEqual(mpi_stability, 0.0)
35
36 def test_mpi_performance (self):
37    """Test MPI performance on the cluster"""
38    # Run the test
39    mpi_performance = self.run_mpi_performance_test()
40    # Check the results
41    self.assertEqual(mpi_performance, 0.0)
42
43 def test_mpi_configuration (self):
44    """Test MPI configuration on the cluster"""
45    # Run the test
46    mpi_configuration = self.run_mpi_configuration_test()
47    # Check the results
48    self.assertEqual(mpi_configuration, 0.0)
49
50 def test_mpi_documentation (self):
51    """Test MPI documentation on the cluster"""
52    # Run the test
53    mpi_documentation = self.run_mpi_documentation_test()
54    # Check the results
55    self.assertEqual(mpi_documentation, 0.0)
56
57 def test_mpi_help (self):
58    """Test MPI help on the cluster"""
59    # Run the test
60    mpi_help = self.run_mpi_help_test()
61    # Check the results
62    self.assertEqual(mpi_help, 0.0)
63
64 def test_mpi_version (self):
65    """Test MPI version on the cluster"""
66    # Run the test
67    mpi_version = self.run_mpi_version_test()
68    # Check the results
69    self.assertEqual(mpi_version, 0.0)
70
71 def test_mpi_license (self):
72    """Test MPI license on the cluster"""
73    # Run the test
74    mpi_license = self.run_mpi_license_test()
75    # Check the results
76    self.assertEqual(mpi_license, 0.0)
77
78 def test_mpi_copyright (self):
79    """Test MPI copyright on the cluster"""
80    # Run the test
81    mpi_copyright = self.run_mpi_copyright_test()
82    # Check the results
83    self.assertEqual(mpi_copyright, 0.0)
84
85 def test_mpi_about (self):
86    """Test MPI about on the cluster"""
87    # Run the test
88    mpi_about = self.run_mpi_about_test()
89    # Check the results
90    self.assertEqual(mpi_about, 0.0)
91
92 def test_mpi_usage (self):
93    """Test MPI usage on the cluster"""
94    # Run the test
95    mpi_usage = self.run_mpi_usage_test()
96    # Check the results
97    self.assertEqual(mpi_usage, 0.0)
98
99 def test_mpi_examples (self):
100   """Test MPI examples on the cluster"""
101   # Run the test
102   mpi_examples = self.run_mpi_examples_test()
103   # Check the results
104   self.assertEqual(mpi_examples, 0.0)
105
106 def test_mpi_tutorial (self):
107   """Test MPI tutorial on the cluster"""
108   # Run the test
109   mpi_tutorial = self.run_mpi_tutorial_test()
110   # Check the results
111   self.assertEqual(mpi_tutorial, 0.0)
112
113 def test_mpi_faq (self):
114   """Test MPI FAQ on the cluster"""
115   # Run the test
116   mpi_faq = self.run_mpi_faq_test()
117   # Check the results
118   self.assertEqual(mpi_faq, 0.0)
119
120 def test_mpi_contact (self):
121   """Test MPI contact on the cluster"""
122   # Run the test
123   mpi_contact = self.run_mpi_contact_test()
124   # Check the results
125   self.assertEqual(mpi_contact, 0.0)
126
127 def test_mpi_feedback (self):
128   """Test MPI feedback on the cluster"""
129   # Run the test
130   mpi_feedback = self.run_mpi_feedback_test()
131   # Check the results
132   self.assertEqual(mpi_feedback, 0.0)
133
134 def test_mpi_privacy (self):
135   """Test MPI privacy on the cluster"""
136   # Run the test
137   mpi_privacy = self.run_mpi_privacy_test()
138   # Check the results
139   self.assertEqual(mpi_privacy, 0.0)
140
141 def test_mpi_terms (self):
142   """Test MPI terms on the cluster"""
143   # Run the test
144   mpi_terms = self.run_mpi_terms_test()
145   # Check the results
146   self.assertEqual(mpi_terms, 0.0)
147
148 def test_mpi_disclaimer (self):
149   """Test MPI disclaimer on the cluster"""
150   # Run the test
151   mpi_disclaimer = self.run_mpi_disclaimer_test()
152   # Check the results
153   self.assertEqual(mpi_disclaimer, 0.0)
154
155 def test_mpi_warranty (self):
156   """Test MPI warranty on the cluster"""
157   # Run the test
158   mpi_warranty = self.run_mpi_warranty_test()
159   # Check the results
160   self.assertEqual(mpi_warranty, 0.0)
161
162 def test_mpi_limitation (self):
163   """Test MPI limitation on the cluster"""
164   # Run the test
165   mpi_limitation = self.run_mpi_limitation_test()
166   # Check the results
167   self.assertEqual(mpi_limitation, 0.0)
168
169 def test_mpi_exception (self):
170   """Test MPI exception on the cluster"""
171   # Run the test
172   mpi_exception = self.run_mpi_exception_test()
173   # Check the results
174   self.assertEqual(mpi_exception, 0.0)
175
176 def test_mpi_notice (self):
177   """Test MPI notice on the cluster"""
178   # Run the test
179   mpi_notice = self.run_mpi_notice_test()
180   # Check the results
181   self.assertEqual(mpi_notice, 0.0)
182
183 def test_mpi_acknowledgment (self):
184   """Test MPI acknowledgment on the cluster"""
185   # Run the test
186   mpi_acknowledgment = self.run_mpi_acknowledgment_test()
187   # Check the results
188   self.assertEqual(mpi_acknowledgment, 0.0)
189
190 def test_mpi_attribution (self):
191   """Test MPI attribution on the cluster"""
192   # Run the test
193   mpi_attribution = self.run_mpi_attribution_test()
194   # Check the results
195   self.assertEqual(mpi_attribution, 0.0)
196
197 def test_mpi_citation (self):
198   """Test MPI citation on the cluster"""
199   # Run the test
200   mpi_citation = self.run_mpi_citation_test()
201   # Check the results
202   self.assertEqual(mpi_citation, 0.0)
203
204 def test_mpi_references (self):
205   """Test MPI references on the cluster"""
206   # Run the test
207   mpi_references = self.run_mpi_references_test()
208   # Check the results
209   self.assertEqual(mpi_references, 0.0)
210
211 def test_mpi_bibliography (self):
212   """Test MPI bibliography on the cluster"""
213   # Run the test
214   mpi_bibliography = self.run_mpi_bibliography_test()
215   # Check the results
216   self.assertEqual(mpi_bibliography, 0.0)
217
218 def test_mpi_index (self):
219   """Test MPI index on the cluster"""
220   # Run the test
221   mpi_index = self.run_mpi_index_test()
222   # Check the results
223   self.assertEqual(mpi_index, 0.0)
224
225 def test_mpi_glossary (self):
226   """Test MPI glossary on the cluster"""
227   # Run the test
228   mpi_glossary = self.run_mpi_glossary_test()
229   # Check the results
230   self.assertEqual(mpi_glossary, 0.0)
231
232 def test_mpi_appendix (self):
233   """Test MPI appendix on the cluster"""
234   # Run the test
235   mpi_appendix = self.run_mpi_appendix_test()
236   # Check the results
237   self.assertEqual(mpi_appendix, 0.0)
238
239 def test_mpi_index_1 (self):
240   """Test MPI index 1 on the cluster"""
241   # Run the test
242   mpi_index_1 = self.run_mpi_index_1_test()
243   # Check the results
244   self.assertEqual(mpi_index_1, 0.0)
245
246 def test_mpi_index_2 (self):
247   """Test MPI index 2 on the cluster"""
248   # Run the test
249   mpi_index_2 = self.run_mpi_index_2_test()
250   # Check the results
251   self.assertEqual(mpi_index_2, 0.0)
252
24

```



# Abstract & Introduction

## Abstract

- This report details a ReFrame-based regression testing suite for MPI intranode and internode communication performance on ULHPC clusters (Aion, Iris).
- Utilizes OSU MicroBenchmarks (osu\_latency, osu\_bw) for point-to-point measurements.
- Evaluated four process placement scenarios and three binary sourcing methods (Source, EasyBuild, EESSI).
- Presents latest performance results, including several instances where performance **deviated from expectations**, and critically examines the baseline reference values used in ReFrame for future regression testing.

## Introduction

- **Goal:** Develop a robust regression testing suite to monitor MPI latency and bandwidth, detecting performance variations due to software/hardware changes or configuration drift.
- **Importance:** Consistent MPI performance is critical for HPC application scalability and efficiency.



# Benchmarks & Core Aim



## Benchmarks Used

- osu\_latency: Measures small message latency
- osu\_bw: Measures large message bandwidth

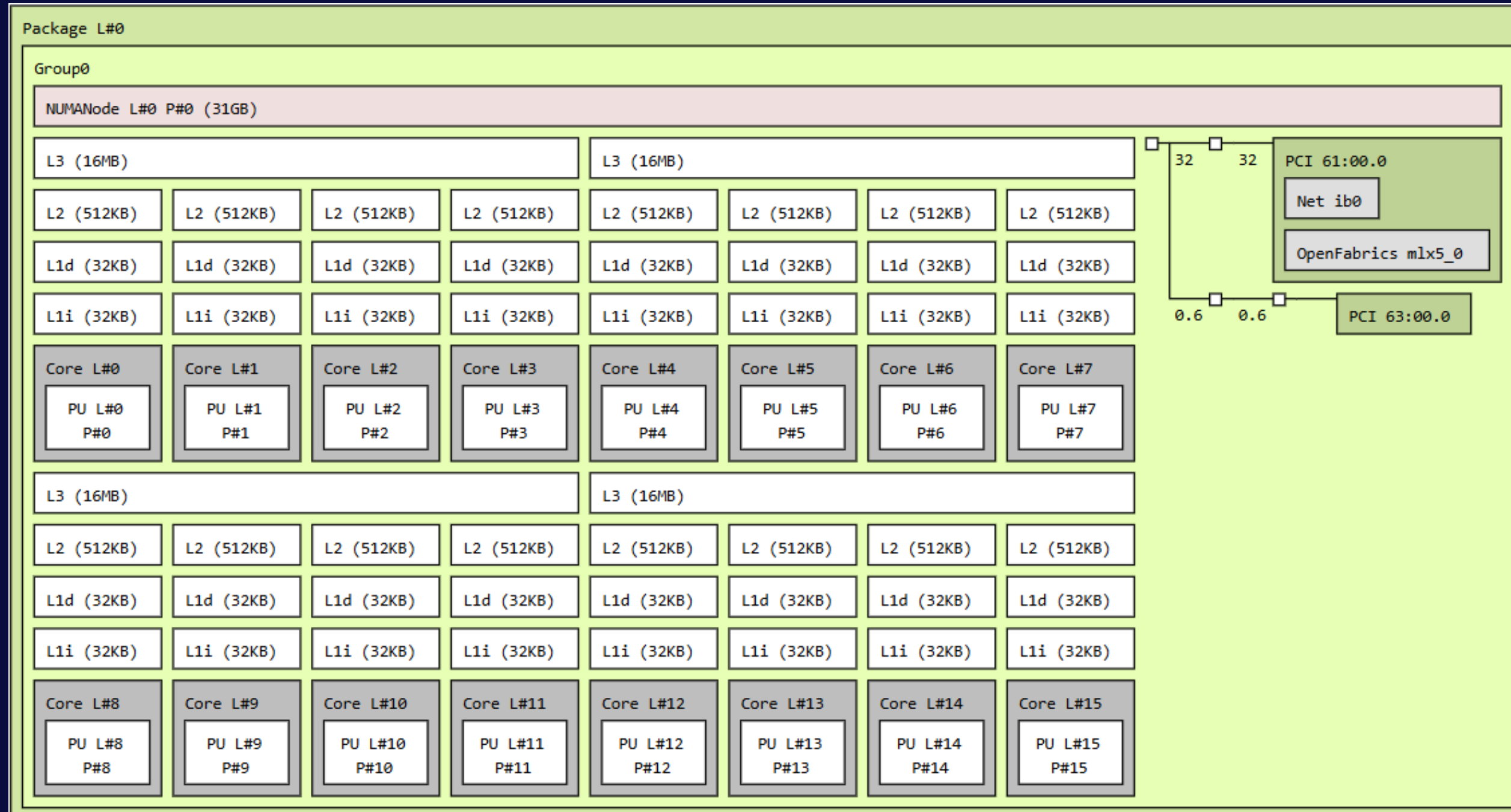


## Core Aim

Compare current data to references to spot regressions or improvements.

ReFrame scripts contain reference values; measured performance is compared, **determining whether performance aligned with expectations or warranted further review.** The results.txt file, reflecting latest runs, now indicates **various performance outcomes.**

# System Architecture (AION)





# Methodology - Test Design

## Target Systems

- **Aion:** AMD EPYC based cluster.
- **Iris:** Intel Skylake based cluster (regular CPU nodes).
  - *Note:*  
*SameSocketDifferentNuma scenario is not applicable/tested on Iris.*

## Benchmarks & Messages

- **osu\_latency:**
  - Message Size: 8192 bytes (typical small message latency).
  - Iterations: 100 warm-up, 1000 measurement.
- **osu\_bw:**
  - Message Size: 1,048,576 bytes (1MB) (peak bandwidth scenario).
  - Iterations: 100 warm-up, 1000 measurement.

## Process Placement

### Process Placement Scenarios (2 MPI processes):

1. **Same NUMA Node (SameNumaNode):** Processes on cores within the same NUMA node, memory local. (Best-case intra-node)
2. **Same Socket, Different NUMA Nodes (SameSocketDifferentNuma):** (Aion-specific) Processes on same socket, different NUMA nodes.
3. **Different Sockets (DifferentSockets):** Processes on same compute node, different physical sockets.
4. **Different Nodes (DifferentNodes):** Processes on different physical compute nodes. (Inter-node)



# Methodology - Compilation & ReFrame

## Implementation

OSU MicroBenchmarks (Version 7.2)

Sourcing:

1. Source Compilation (SOURCE): Compiled directly from source using foss/2023b toolchain.
2. EasyBuild Compilation (EASYBUILD): Compiled using an EasyBuild recipe with foss/2023b toolchain.
3. EESSI Binaries (EESSI): Pre-compiled binaries from EESSI software stack (OSU-Micro-Benchmarks/7.2-gompi-2023b).

### ReFrame Implementation:

- Regression tests implemented using the ReFrame framework.
- Specific SLURM options and OpenMPI MCA parameters enforced process placement.
- Performance extracted by parsing benchmark output.
- **Crucially:** ReFrame scripts contain reference values. Measured performance is compared against these, **determining if the measurement is within the expected range or if it signals a deviation.**
  - The results.txt file, reflecting the latest runs, shows a **variety of performance outcomes relative to these references**, forming the basis of this analysis.





# Results Overview: NUMA Performance Benchmarks

## Results – Overview & General Trends

### General Performance Trends Observed:

#### Bandwidth (Higher is Better):

- Typically highest for SameNumaNode (with exceptions on Iris).
- Decreases for DifferentSockets, SameSocketDifferentNuma (Aion), and lowest for DifferentNodes.

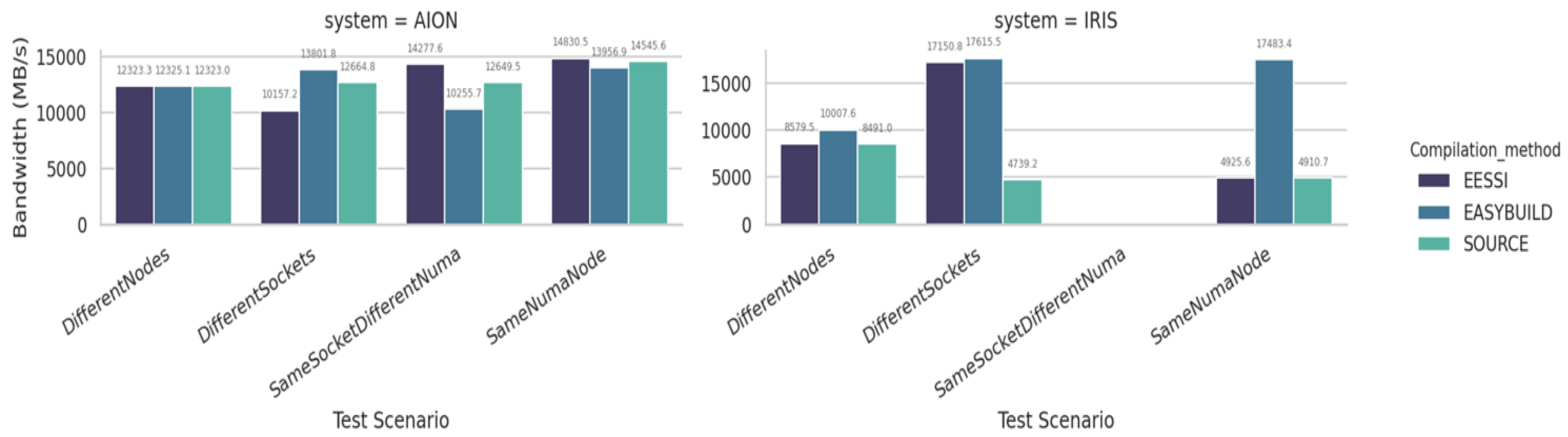
#### Latency (Lower is Better):

- Generally lowest for SameNumaNode.
  - *Note: These often indicated performance better than very strict reference values, thus highlighting them for review.*
- Increases as communication crosses NUMA, socket, and inter-node boundaries.

**Latest results.txt shows a variety of performance outcomes relative to reference values.**

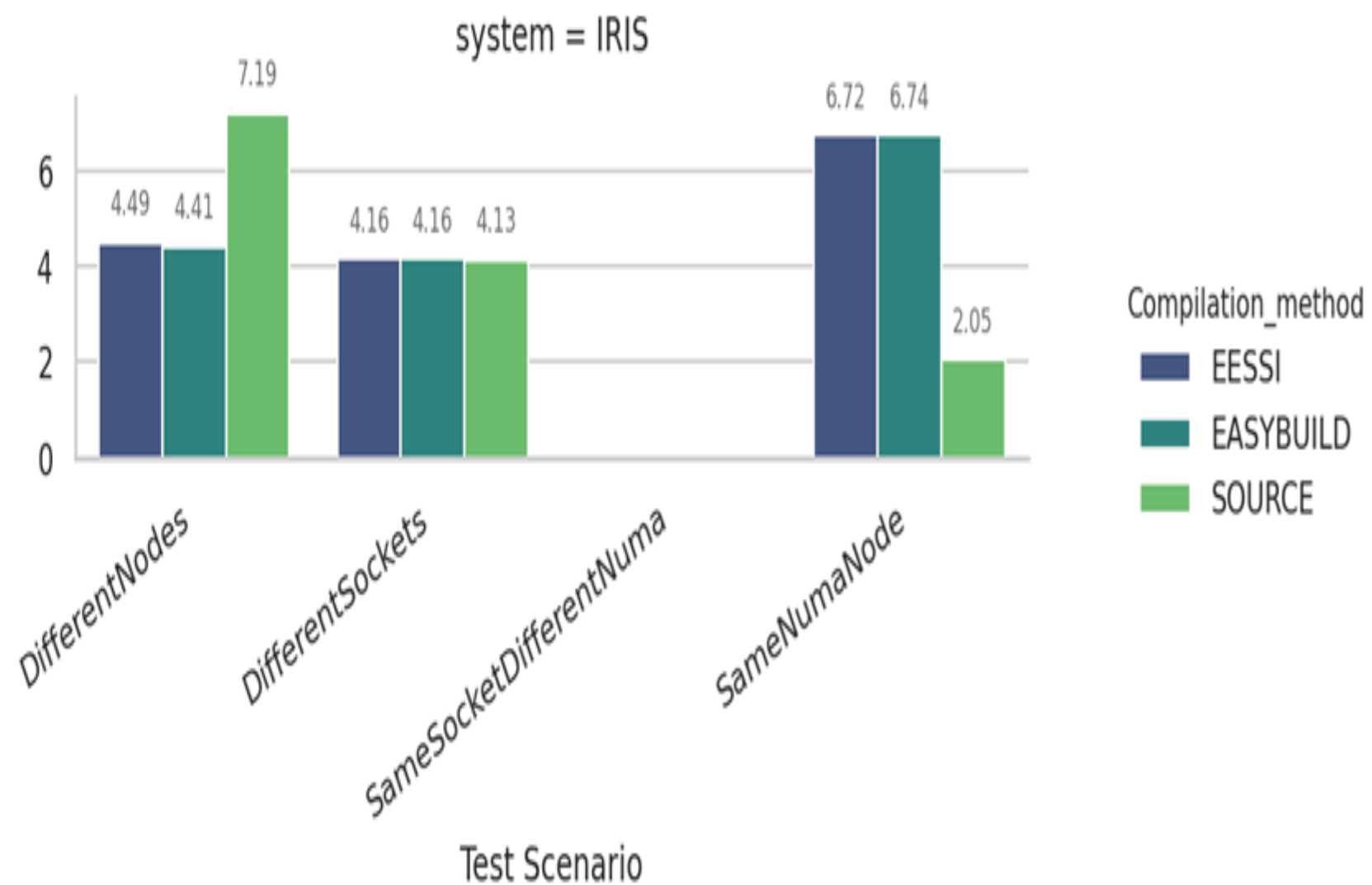
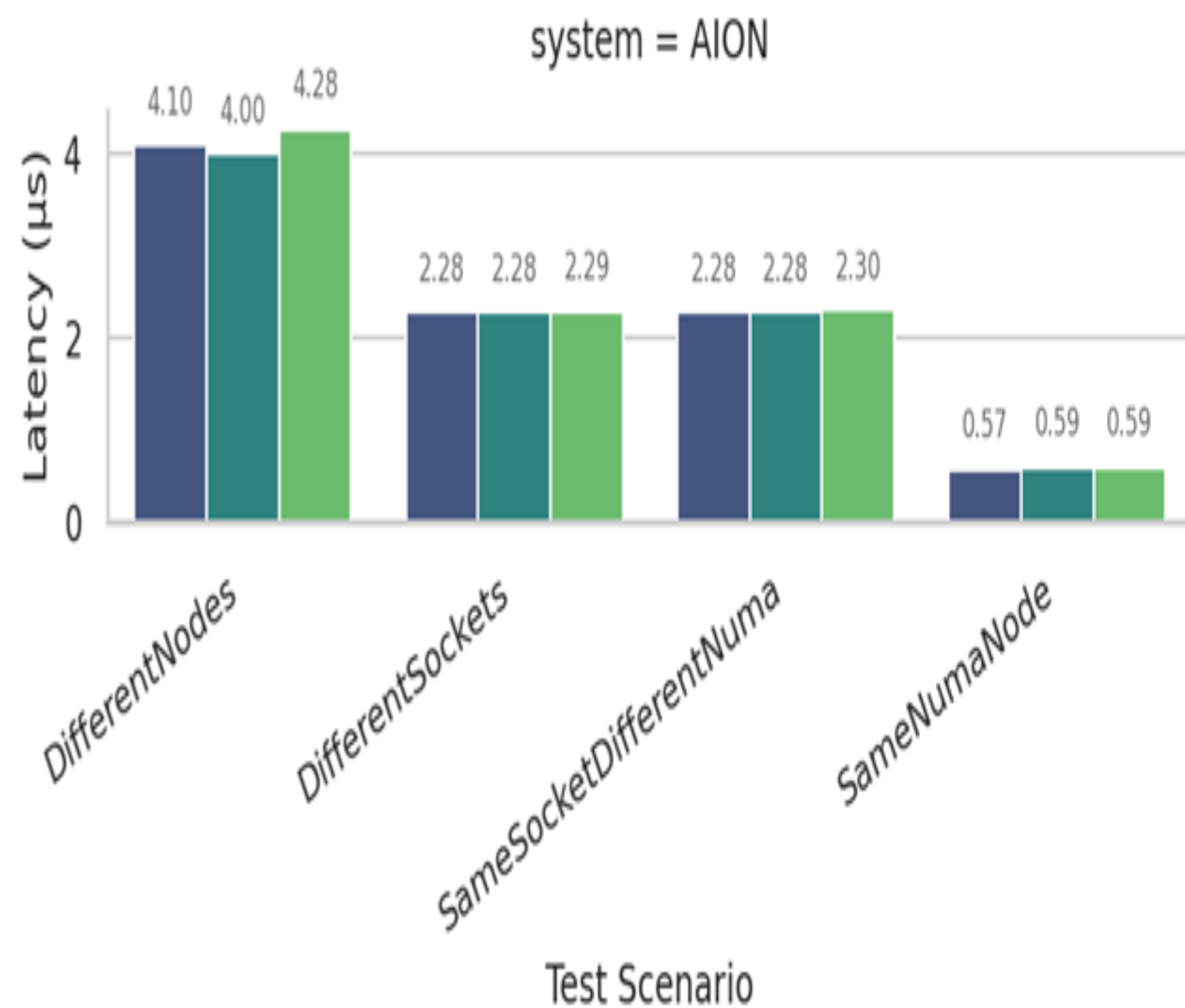
- This presentation will analyze observed values and critically discuss the appropriateness of the underlying ReFrame reference values that led to these outcomes.

# MPI Bandwidth Comparison (Higher is Better)





# MPI Latency Comparison (Lower is Better)

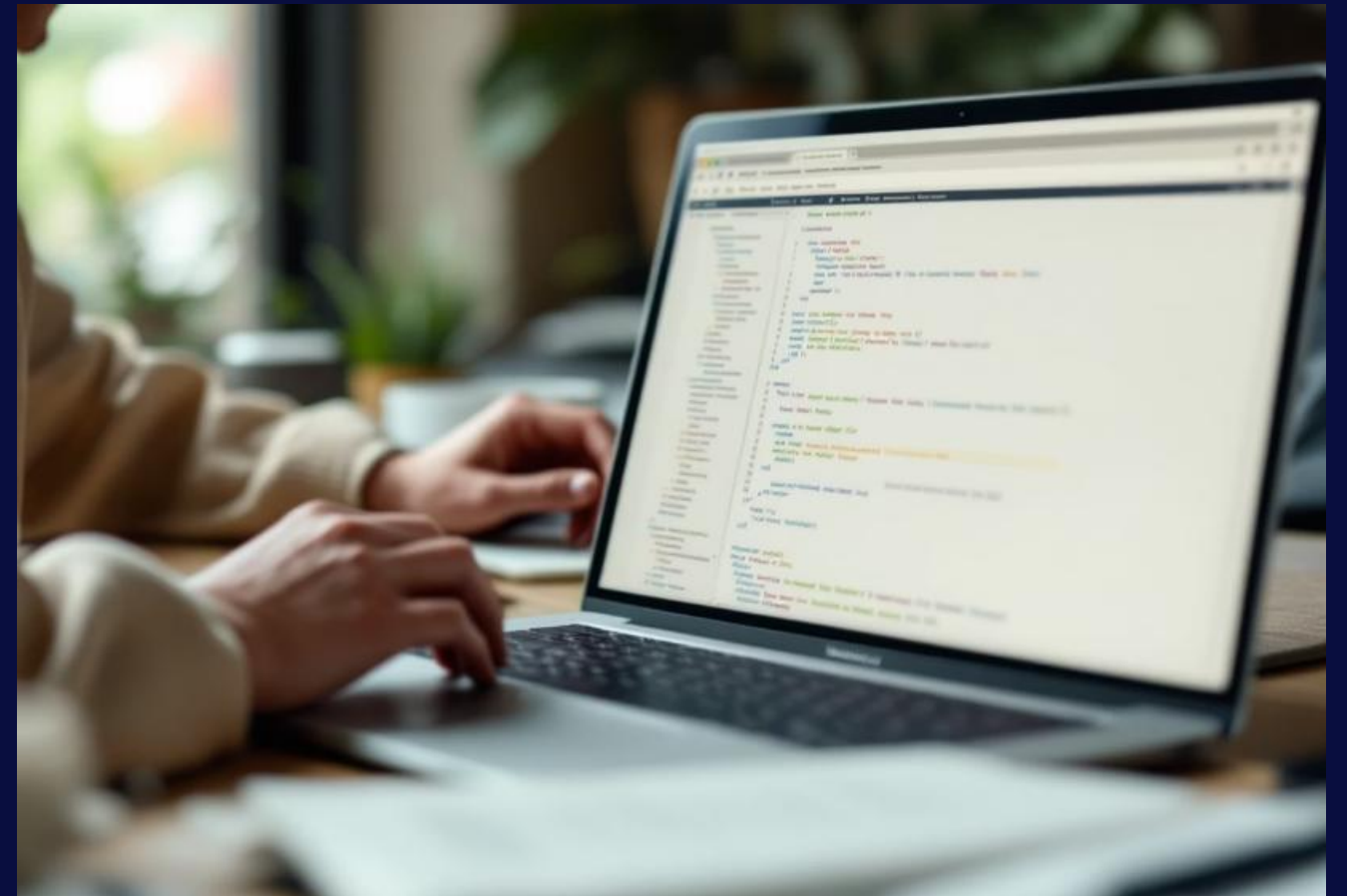


# Reproducibility of Regression Testing

All ReFrame tests, configuration files, EasyBuild recipe, and report are stored in a Git repository:

<https://github.com/PaulAroo/Regression-testing>

Instructions for cloning, loading modules, and running tests are detailed in the [README.md](#).







THANK YOU

