Modeling Non-Determinism in HPC Applications

Dylan Chapp
Advisor: Michela Taufer

THE UNIVERSITY OF TENNESSEE
KNOXVILLE
BIG ORANGE. BIG IDEAS.®

Non-Determinism and Correctness in HPC

- **HPC Community Position:** "Non-determinism control" and "anomaly detection" identified in DOE report on the 2017 HPC Correctness Summit as key challenges in bug detection and localization [1]

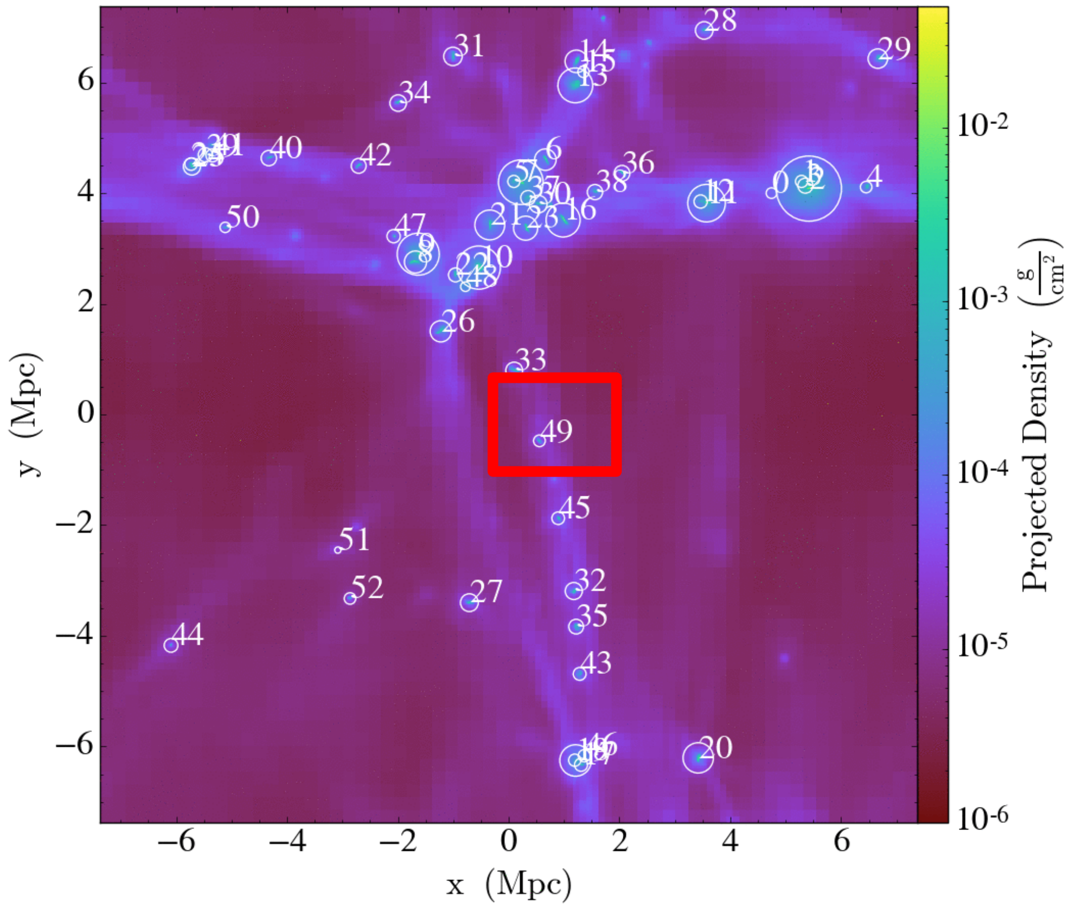
1. Gopalakrishnan, G., Hovland, P.D., Iancu, C., Krishnamoorthy, S., Laguna, I., Lethin, R.A., Sen, K., Siegel, S.F. and Solar-Lezama, A., 2017. Report of the HPC Correctness Summit, Jan 25--26, 2017, Washington, DC. *arXiv preprint arXiv:1705.07478*. See section 3.2.4, 3.2.5

Non-Determinism and Correctness in HPC

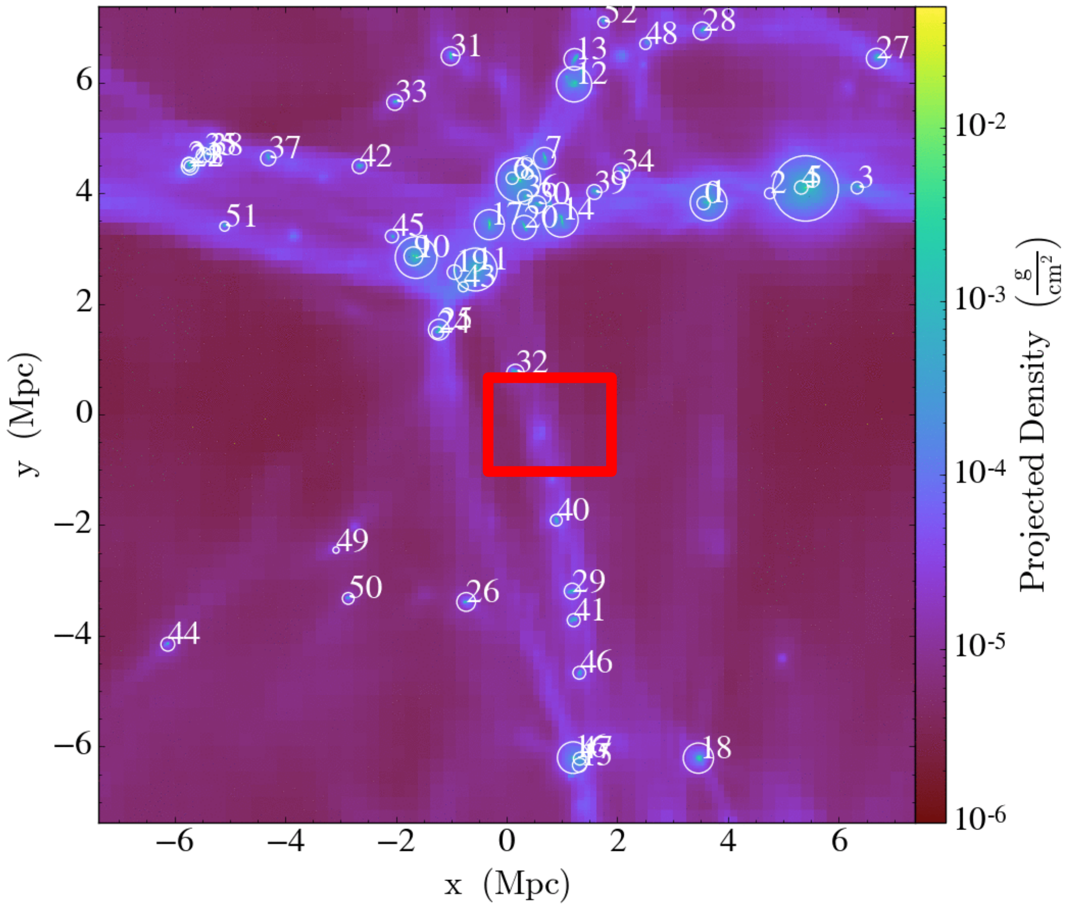
- **HPC Community Position:** "Non-determinism control" and "anomaly detection" identified in DOE report on the 2017 HPC Correctness Summit as key challenges in bug detection and localization [1]
- **Our Position:**
  - These challenges go hand-in-hand.
  - Detecting when and how applications act non-deterministically **in anomalous ways** is critical

1. Gopalakrishnan, G., Hovland, P.D., Iancu, C., Krishnamoorthy, S., Laguna, I., Lethin, R.A., Sen, K., Siegel, S.F. and Solar-Lezama, A., 2017. Report of the HPC Correctness Summit, Jan 25--26, 2017, Washington, DC. *arXiv preprint arXiv:1705.07478*. See section 3.2.4, 3.2.5

Impacts of Non-Determinism on Scientific Outcomes



Run where galactic halo **was detected**



Run where galactic halo was **not detected**

1. Stodden, V. and Krafczyk, M.S., 2018. Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context.

Interaction Between Non-Associativity and Non-Determinism

$a = 10^9,\ b = -10^9,\ c = 10^{-9}$

Summation order 1

$(a + b) + c = (10^9 - 10^9) + 10^{-9} = 10^{-9}$

Summation order 2

$a + (b + c) = 10^9 + (-10^9 + 10^{-9}) = 0$

Interaction Between Non-Associativity and Non-Determinism

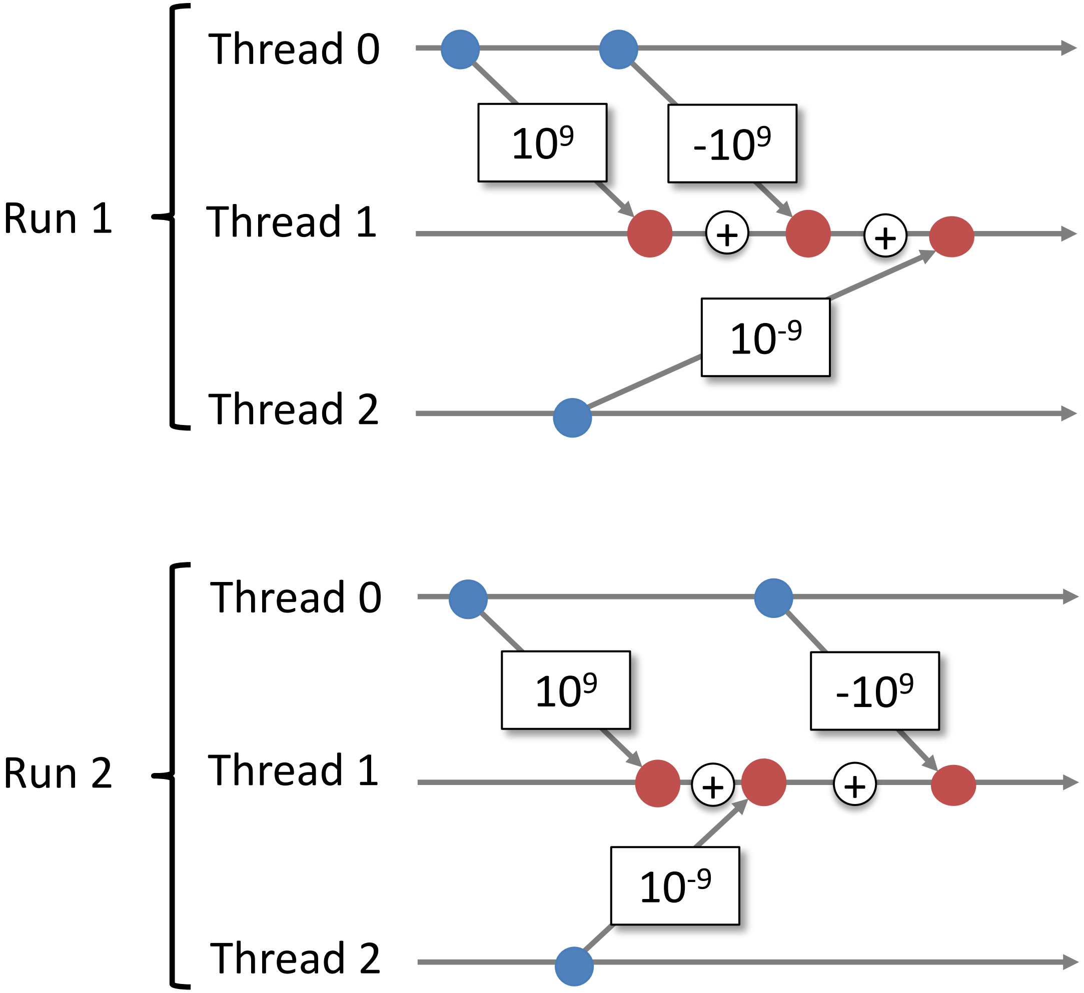
$a = 10^9,\ b = -10^9,\ c = 10^{-9}$

Summation order 1

$(a + b) + c = (10^9 - 10^9) + 10^{-9} = 10^{-9}$

Summation order 2

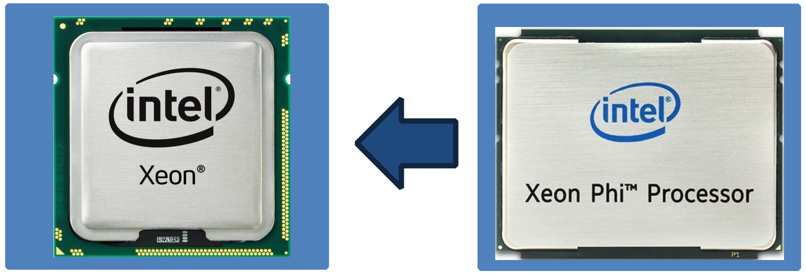
$a + (b + c) = 10^9 + (-10^9 + 10^{-9}) = 0$



Impacts of Non-Determinism on Correctness

Example 1:

- Rounding difference between Xeon CPU and Xeon Phi caused message count to differ on CPU code vs. accelerator code [1]
- Message count difference induced deadlock

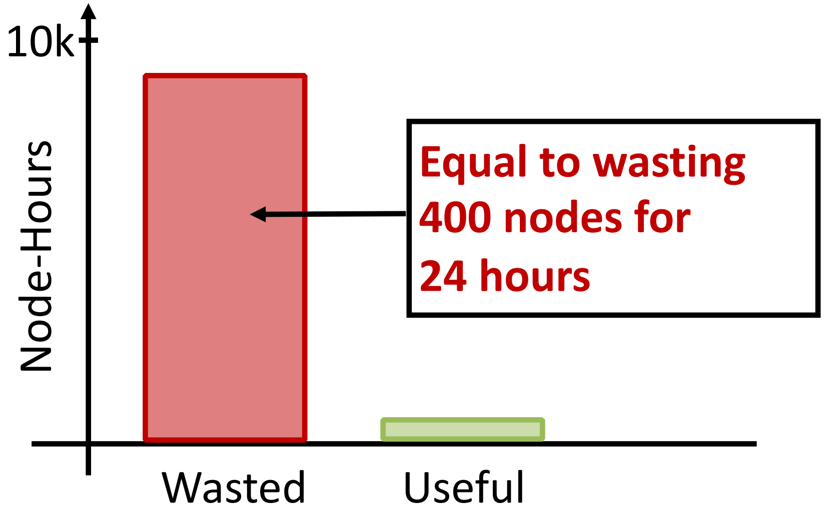


Expecting 62 messages

Sends 63 messages
DEADLOCK

Example 2:

- A non-deterministic bug in Diablo/HYPRE 2.10.1 [2]
- Application hung after several hours, in approximately 1/50 runs
- Cost of debugging effort:
  - **18 months of scientists' time**
  - **9560 node-hours**



1. Gopalakrishnan, G., Hovland, P.D., Iancu, C., Krishnamoorthy, S., Laguna, I., Lethin, R.A., Sen, K., Siegel, S.F. and Solar-Lezama, A., 2017. Report of the HPC Correctness Summit, Jan 25--26, 2017, Washington, DC. *arXiv preprint arXiv:1705.07478*.
2. Sato, K., Ahn, D.H., Laguna, I., Lee, G.L., Schulz, M. and Chambreau, C.M., 2017, January. Noise injection techniques to expose subtle and unintended message races. In *ACM SIGPLAN Notices* (Vol. 52, No. 8, pp. 89-101). ACM.

Summary of Challenges

- HPC applications involve the interaction between:
  - Floating-point non-associativity
  - Communication non-determinism

Summary of Challenges

- HPC applications involve the interaction between:
  - Floating-point non-associativity
  - Communication non-determinism
- Impacts can range from program incorrectness to irreproducibility of scientific outputs

Summary of Challenges

- HPC applications involve the interaction between:
  - Floating-point non-associativity
  - Communication non-determinism
- Impacts can range from program incorrectness to irreproducibility of scientific outputs
- Mitigation strategies exist, but can be costly, limited in scope, and fail to address a key need:

Summary of Challenges

- HPC applications involve the interaction between:
  - Floating-point non-associativity
  - Communication non-determinism
- Impacts can range from program incorrectness to irreproducibility of scientific outputs
- Mitigation strategies exist, but can be costly, limited in scope, and fail to address a key need: **Linking observable non-determinism to its root causes**

Our Approach

- Need to link observations of potentially harmful non-determinism to potential root causes

Our Approach

- Need to link observations of potentially harmful non-determinism to potential root causes
- Need to distinguish between anomalous non-deterministic application behaviors and expected ones

Our Approach

- Need to link observations of potentially harmful non-determinism to potential root causes
- Need to distinguish between anomalous non-deterministic application behaviors and expected ones
- Need a metric for execution similarity

Our Approach

- Need to link observations of potentially harmful non-determinism to potential root causes
- Need to distinguish between anomalous non-deterministic application behaviors and expected ones
- Need a metric for execution similarity
- Need a model of executions that supports such a metric

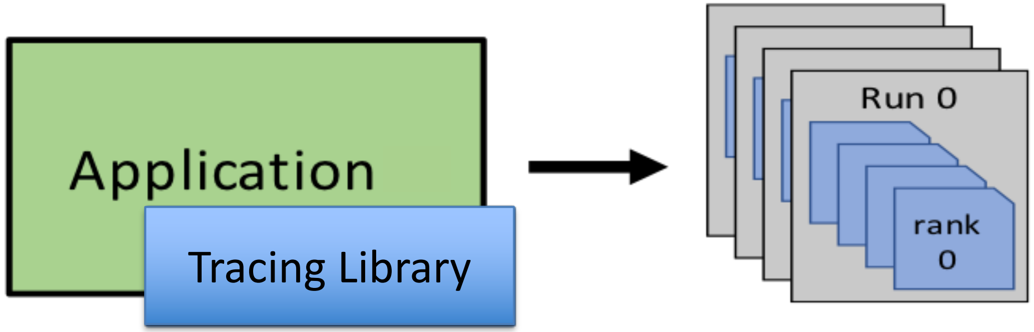
Workflow for Non-Determinism Characterization

- **Phase 1:** Build graph-structured models of executions
- **Phase 2:** Quantify cross-execution trends in non-deterministic communication via graph similarity
- **Phase 3**: Detect periods of anomalous execution dissimilarity and localize potential root causes

Workflow for Non-Determinism Characterization

- **Phase 1:** Build graph-structured models of executions
- **Phase 2:** Quantify cross-execution trends in non-deterministic communication via graph similarity
- **Phase 3**: Detect periods of anomalous execution dissimilarity and localize potential root causes

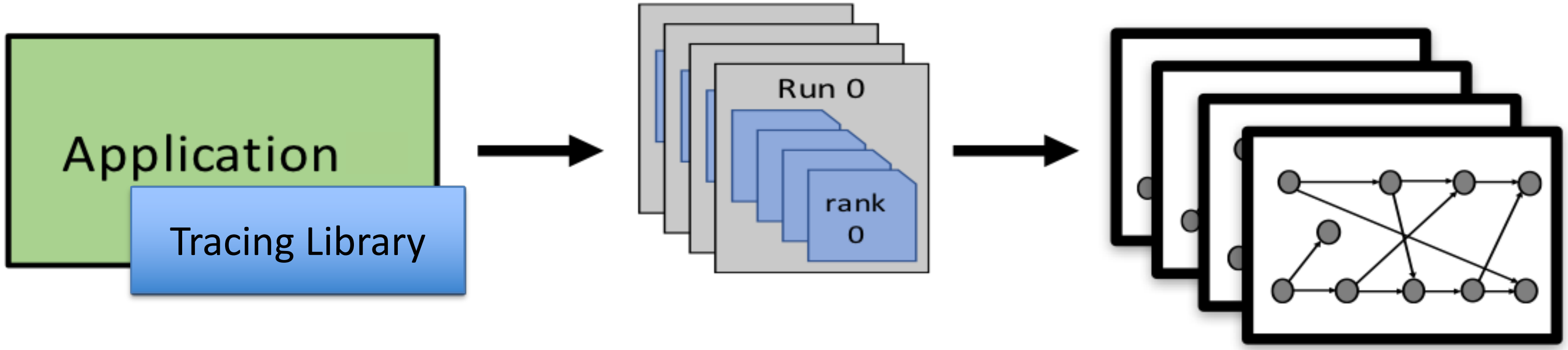
From Traces to Event Graphs



- We trace a non-deterministic application multiple times, capturing a record of communication events

1. Rodrigues, A.F., Voskuilen, G.R., Hammond, S.D. and Hemmert, K.S., 2016. *Structural Simulation Toolkit (SST)* (No. SAND2016-3693PE). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
2. https://github.com/sstsimulator/sst-dumpi

From Traces to Event Graphs

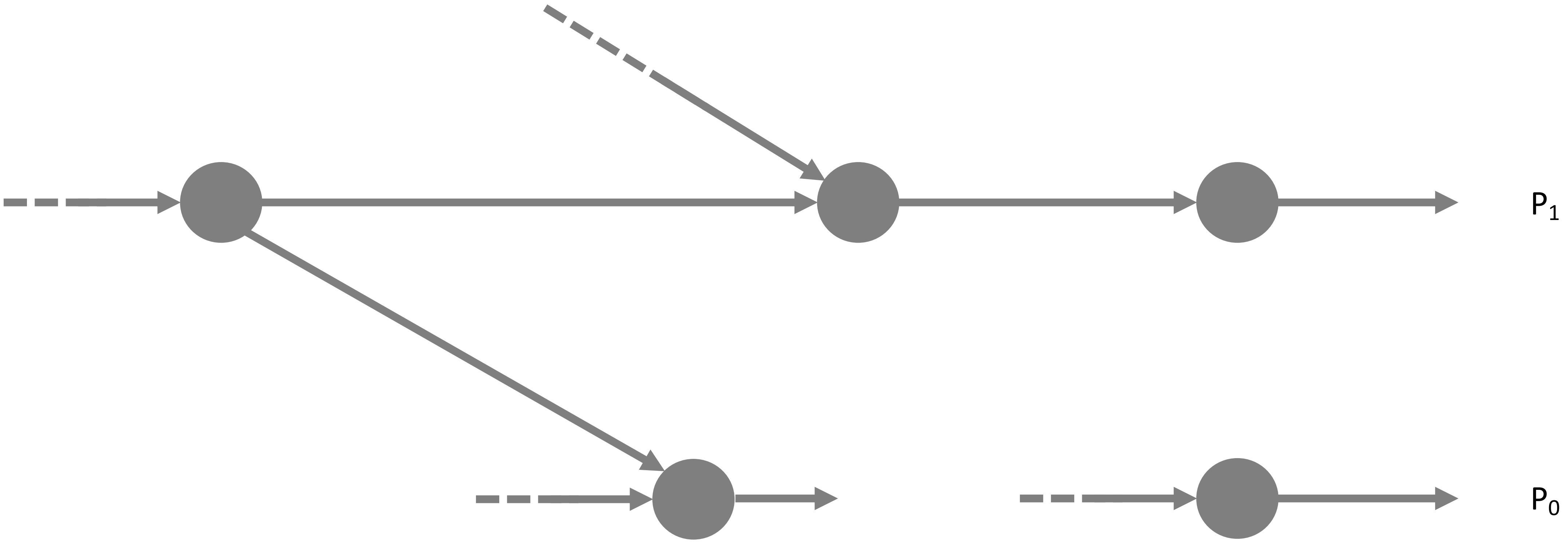


- We trace a non-deterministic application multiple times, capturing a record of communication events
- We convert the set of trace files to a graph-structured model of the inter-process communication that occurred during the execution—i.e., the event graph [3]

1. Rodrigues, A.F., Voskuilen, G.R., Hammond, S.D. and Hemmert, K.S., 2016. *Structural Simulation Toolkit (SST)* (No. SAND2016-3693PE). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
2. https://github.com/sstsimulator/sst-dumpi
3. Kranzlmüller, D., 2000. *Event graph analysis for debugging massively parallel programs*.

Event Graph Structure

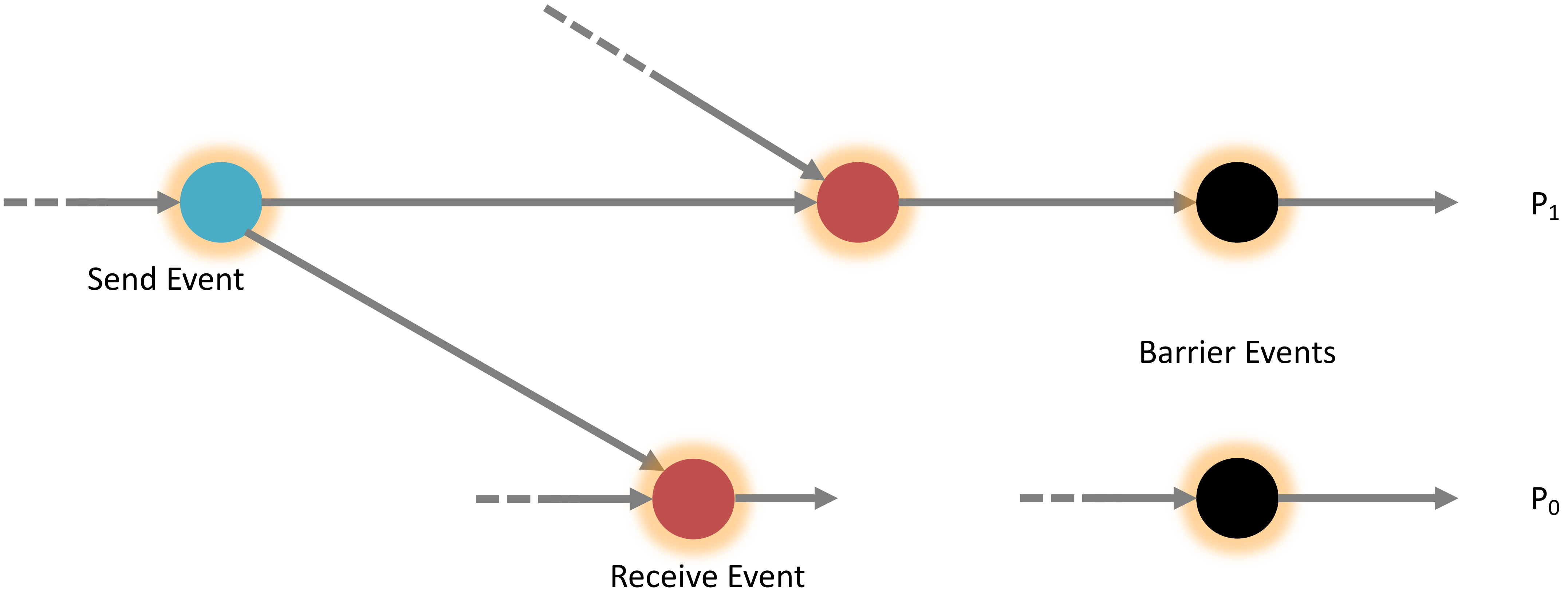
- Directed, acyclic graph (DAG) representing a message-passing program execution [1]



1. Kranzlmüller, D., 2000. *Event graph analysis for debugging massively parallel programs.*

Event Graph Structure

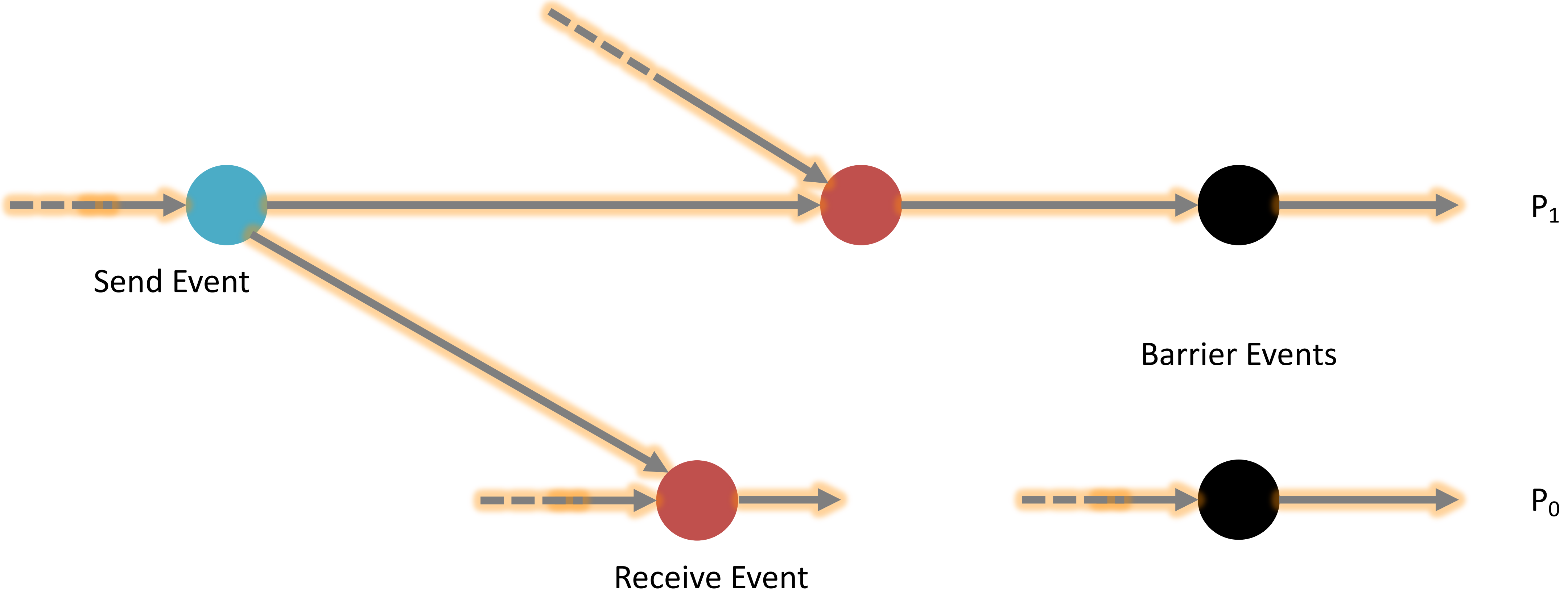
- Directed, acyclic graph (DAG) representing a message-passing program execution [1]
- Vertices represent communication events (e.g., message sends, receives, and barriers)



1. Kranzlmüller, D., 2000. *Event graph analysis for debugging massively parallel programs.*

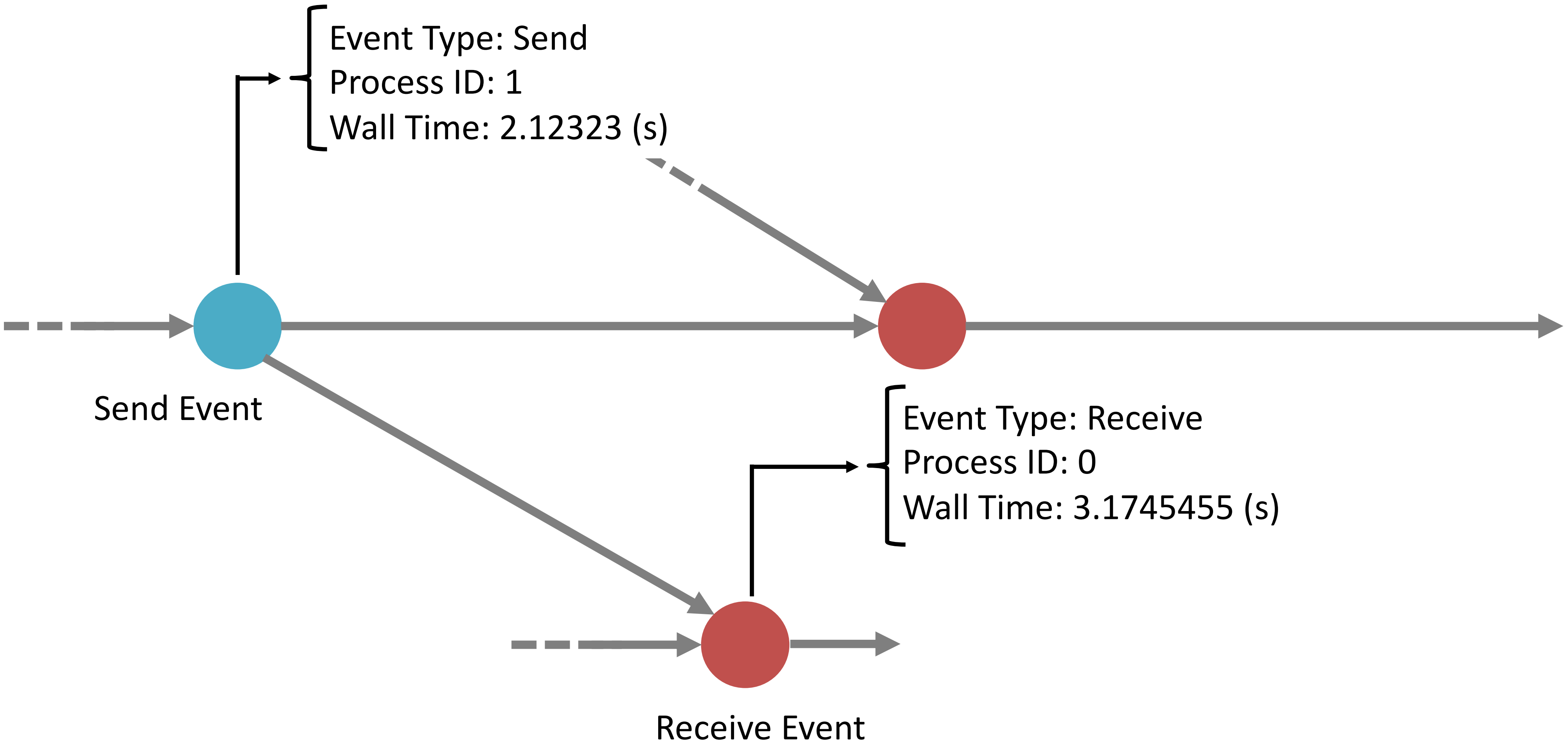
Event Graph Structure

- Directed, acyclic graph (DAG) representing a message-passing program execution [1]
- Vertices represent communication events (e.g., message sends, receives, and barriers)
- Edges represent “happens-before”[2] relationship between events

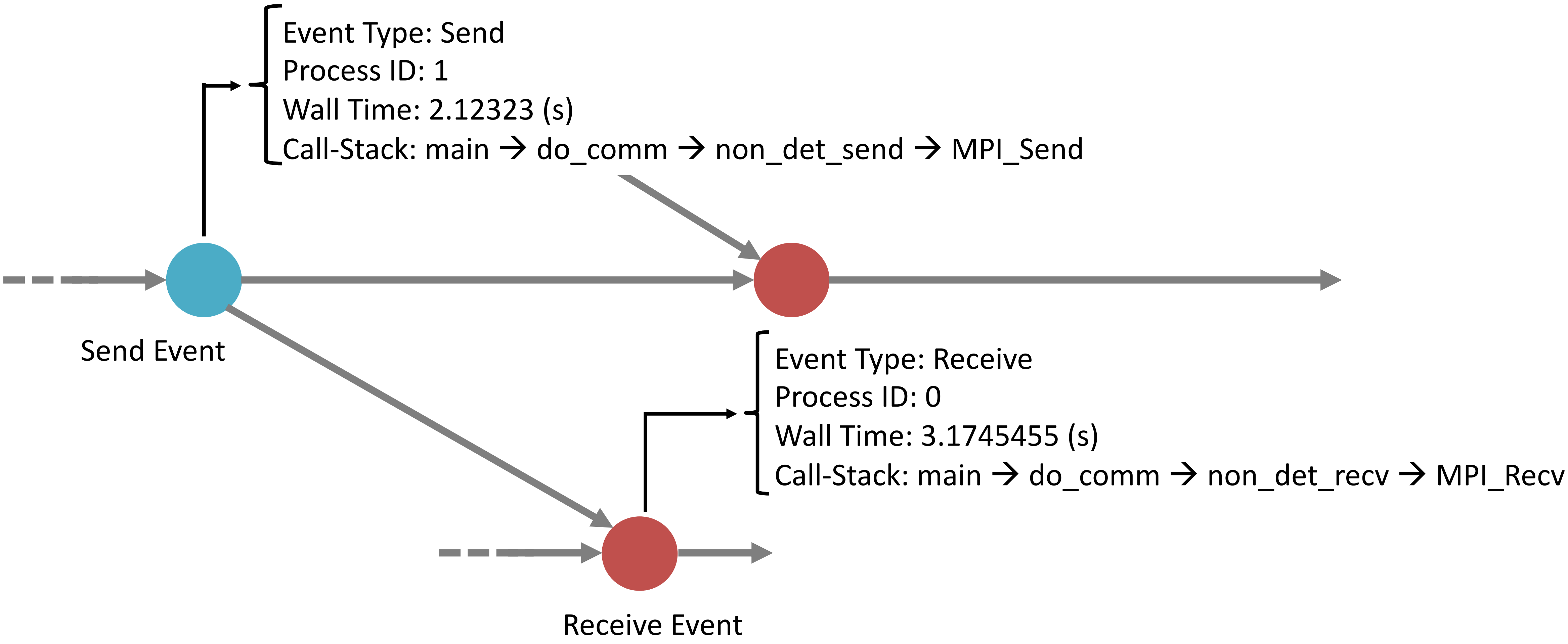


1. Kranzlmüller, D., 2000. *Event graph analysis for debugging massively parallel programs.*
2. Lamport, L., 1978. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, *21*(7), pp.558-565.

Event Graph Vertex Labels

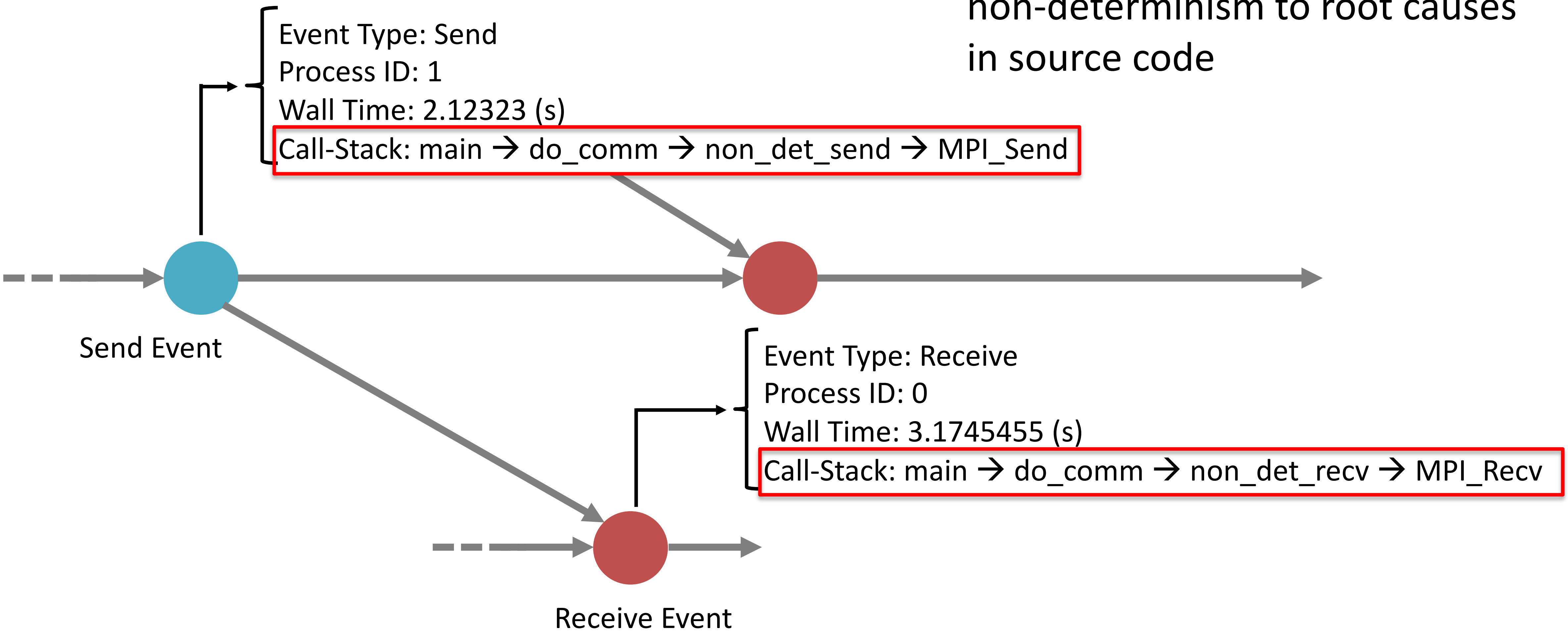


Event Graph Vertex Labels



Event Graph Vertex Labels

Call-stack labels link run-time non-determinism to root causes in source code



Summary of Event Graph Model

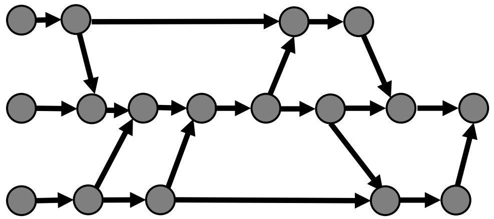
- Traces of MPI application → DAG
- Vertices → communication events
- Edges → happens-before orders
- Vertex Labels:
  - Event type → What happened?
  - Process ID → In which process did it happen?
  - Timestamp → When did it happen?
  - Call-Stack → Where in the code did it come from?

Workflow for Non-Determinism Characterization

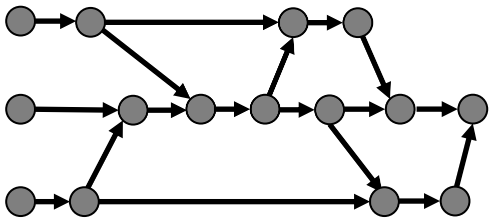
- **Phase 1:** Build graph-structured models of executions
- **Phase 2:** Quantify cross-execution trends in non-deterministic communication via graph similarity
- **Phase 3**: Detect periods of anomalous execution dissimilarity and localize potential root causes

Graph Similarity via Graph Kernels

K (



,

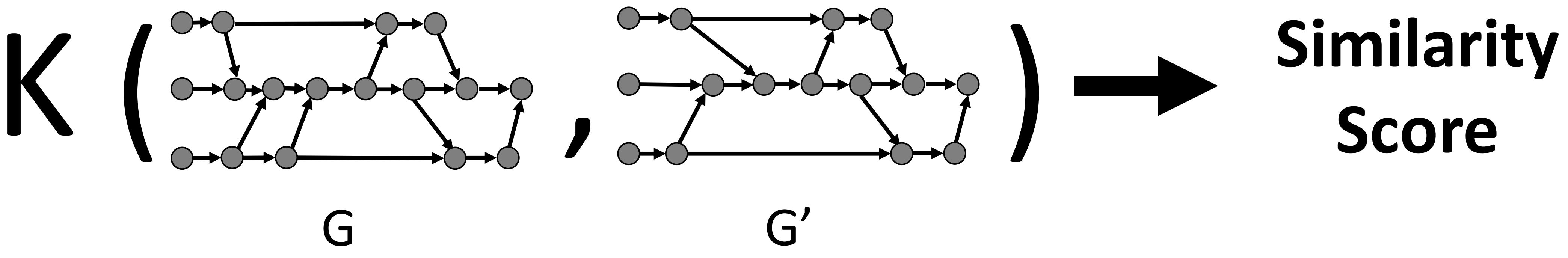


) → **Similarity Score**

G

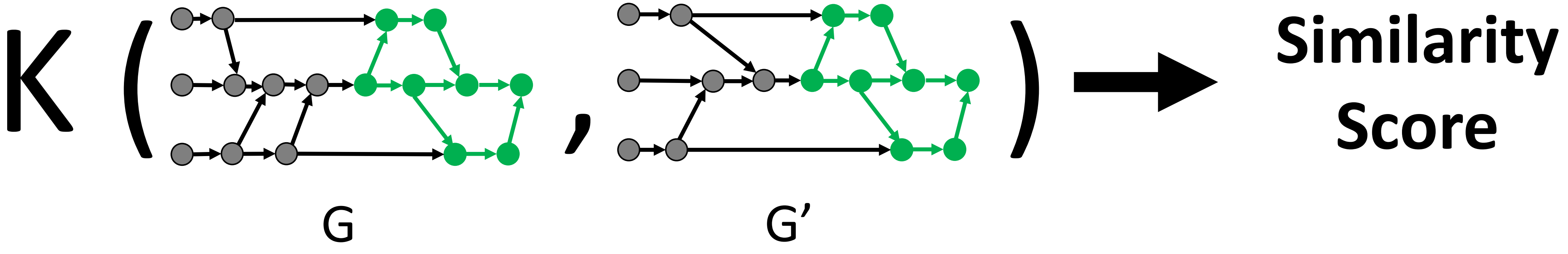
G’

Graph Similarity via Graph Kernels



Intuition: K counts matching substructures

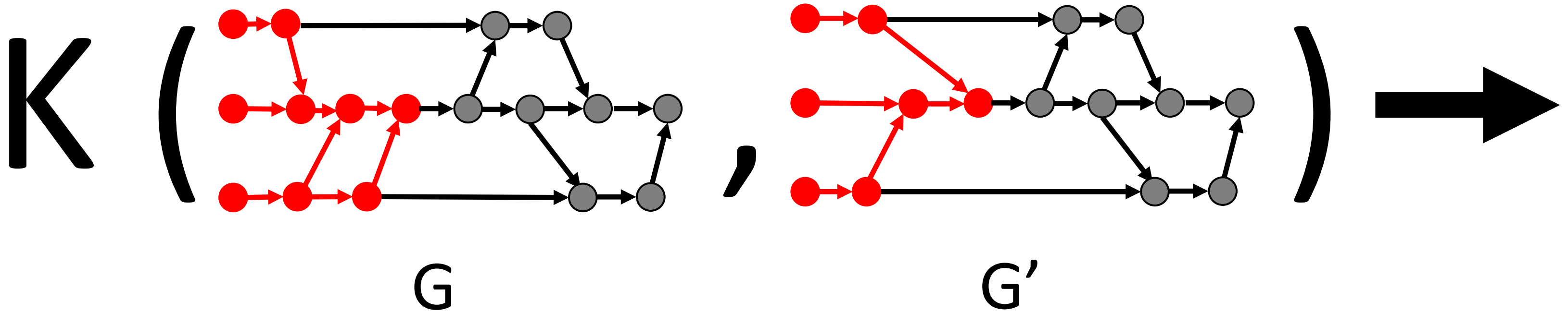
Graph Similarity via Graph Kernels



Intuition: K counts matching substructures

- Matches between G and G' increase score

Graph Similarity via Graph Kernels



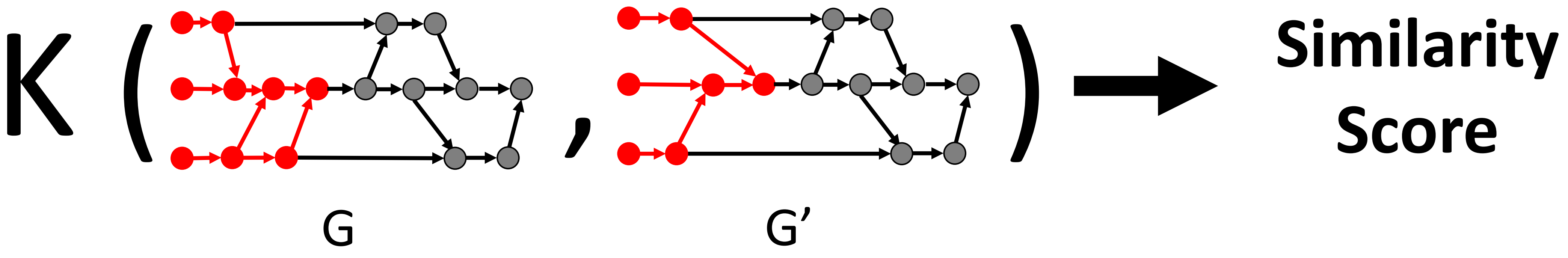
Similarity Score

Intuition: K counts matching substructures

- Matches between G and G’ increase score
- Differences do not

1. Phillips, J.M. and Venkatasubramanian, S., 2011. A gentle introduction to the kernel distance. *arXiv preprint arXiv:1103.1625*.

Graph Similarity via Graph Kernels



Intuition: K counts matching substructures

- Matches between G and G’ increase score
- **Differences** do not
- Graph kernel K **induces a metric** → the graph kernel distance [1]

Formula: $\mathrm{D}(\mathrm{G},\mathrm{G}') = \sqrt{K(G,G') + K(G,G') - 2K(G,G')}$

1. Phillips, J.M. and Venkatasubramanian, S., 2011. A gentle introduction to the kernel distance. *arXiv preprint arXiv:1103.1625*.

Quantifying Non-Determinism via Graph Kernel Distance

We evaluate the Weisfeiler-Lehman Subtree Pattern Kernel for quantifying dissimilarity between event graphs:

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.
2. Yanardag, P. and Vishwanathan, S.V.N., 2015, August. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1365-1374). ACM.
3. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P. and Kundu, M., 2018. The journey of graph kernels through two decades. *Computer Science Review*, *27*, pp.88-111.

Quantifying Non-Determinism via Graph Kernel Distance

We evaluate the Weisfeiler-Lehman Subtree Pattern Kernel for quantifying dissimilarity between event graphs:

- Demonstrated performance on other graph classification tasks [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.
2. Yanardag, P. and Vishwanathan, S.V.N., 2015, August. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1365-1374). ACM.
3. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P. and Kundu, M., 2018. The journey of graph kernels through two decades. *Computer Science Review*, *27*, pp.88-111.

Quantifying Non-Determinism via Graph Kernel Distance

We evaluate the Weisfeiler-Lehman Subtree Pattern Kernel for quantifying dissimilarity between event graphs:

- Demonstrated performance on other graph classification tasks [1]
- Scalability compared to other graph kernels

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.
2. Yanardag, P. and Vishwanathan, S.V.N., 2015, August. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1365-1374). ACM.
3. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P. and Kundu, M., 2018. The journey of graph kernels through two decades. *Computer Science Review*, *27*, pp.88-111.

Quantifying Non-Determinism via Graph Kernel Distance

We evaluate the Weisfeiler-Lehman Subtree Pattern Kernel for quantifying dissimilarity between event graphs:

- Demonstrated performance on other graph classification tasks [1]
- Scalability compared to other graph kernels
- Incorporation of arbitrary vertex label data

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.
2. Yanardag, P. and Vishwanathan, S.V.N., 2015, August. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1365-1374). ACM.
3. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P. and Kundu, M., 2018. The journey of graph kernels through two decades. *Computer Science Review*, *27*, pp.88-111.

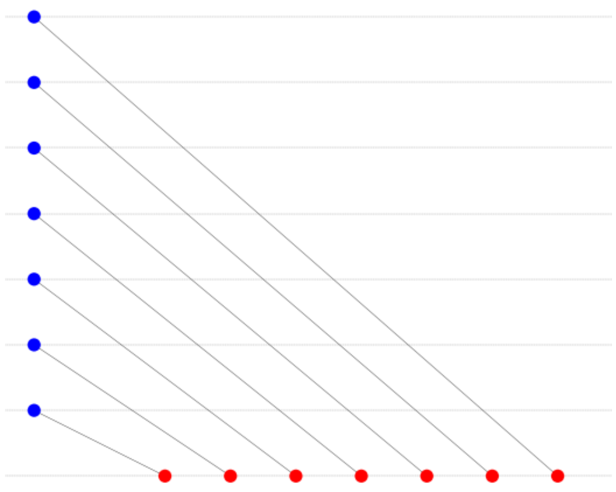
Kernel Distance Evaluation Methodology

- Construct event graphs for common communication patterns with:
  - Controlled degree of non-determinism
  - Fixed amount of communication volume

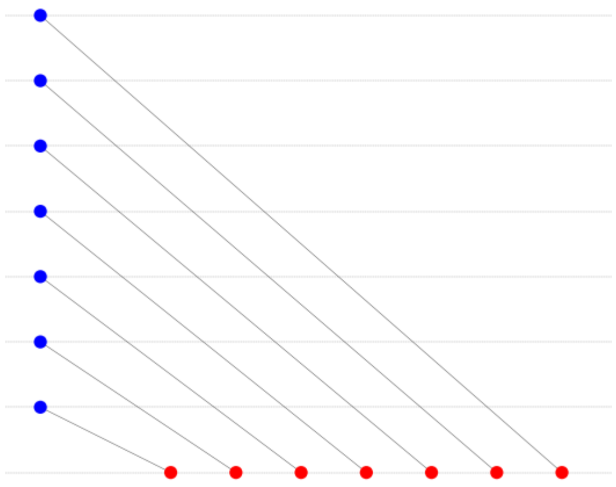
Kernel Distance Evaluation Methodology

- Construct event graphs for common communication patterns with:
  - Controlled degree of non-determinism
  - Fixed amount of communication volume
- **Hypothesis:** As greater non-determinism is permitted in the runs, the graph kernel distances between the event graphs representing those runs will increase

Run 1

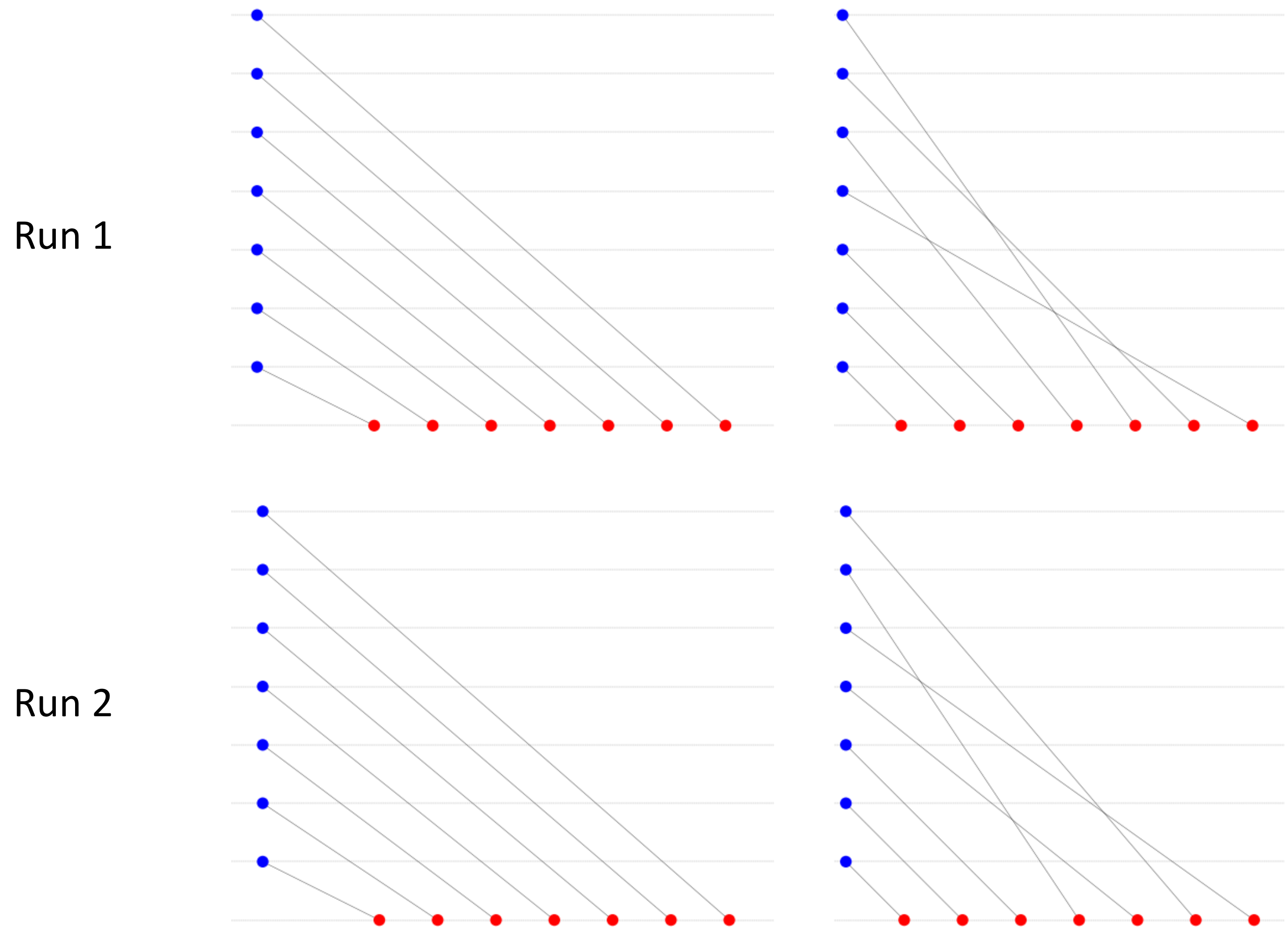


Run 2



Deterministic

All receives guaranteed to occur
in same order run-to-run



Deterministic

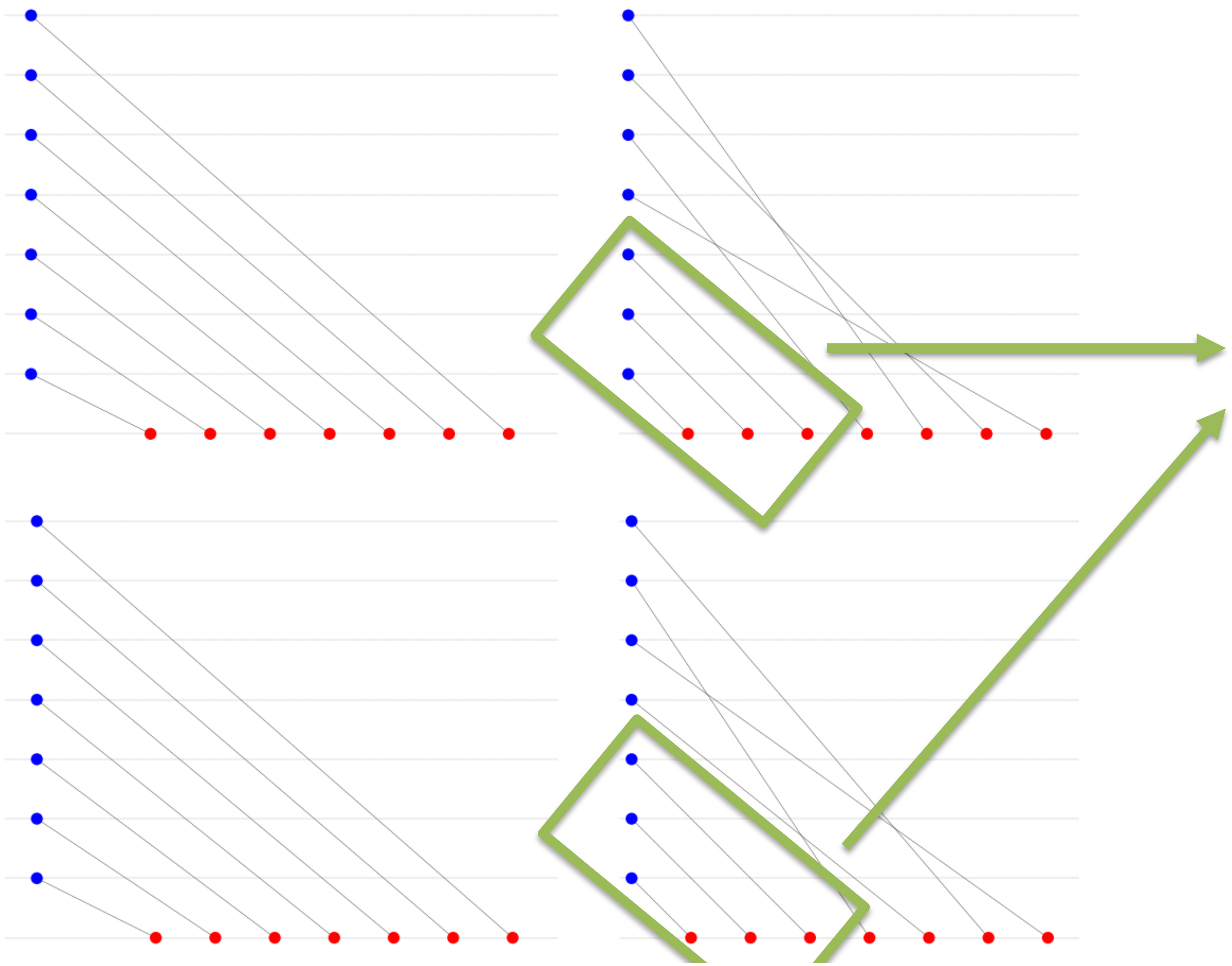
All receives guaranteed to occur in same order run-to-run

Partially Non-Deterministic

Some receives guaranteed to occur in same order

Run 1

Run 2



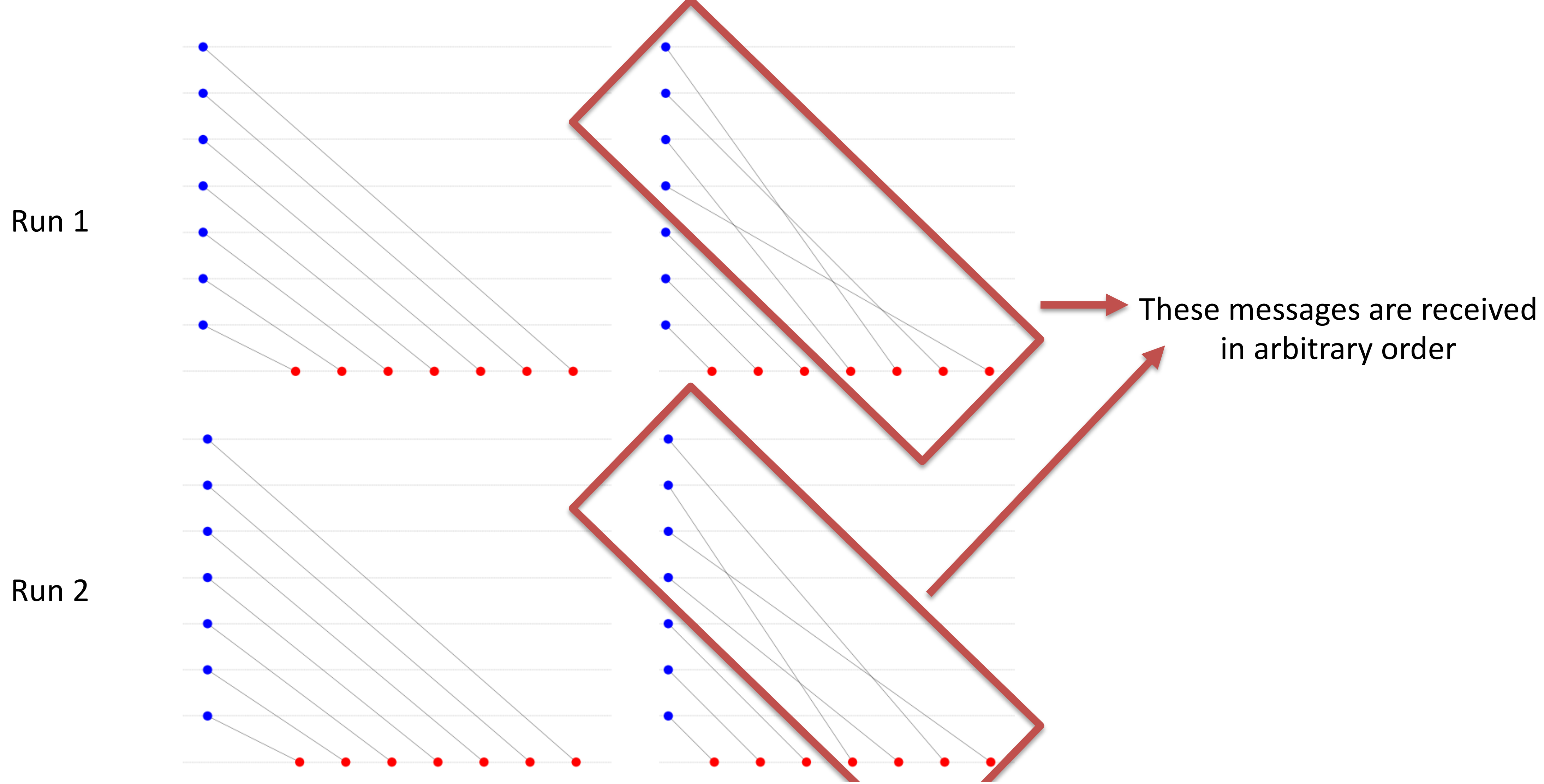
These messages are always received in same order

<u>Deterministic</u>

All receives guaranteed to occur in same order run-to-run

<u>Partially Non-Deterministic</u>

Some receives guaranteed to occur in same order



Deterministic

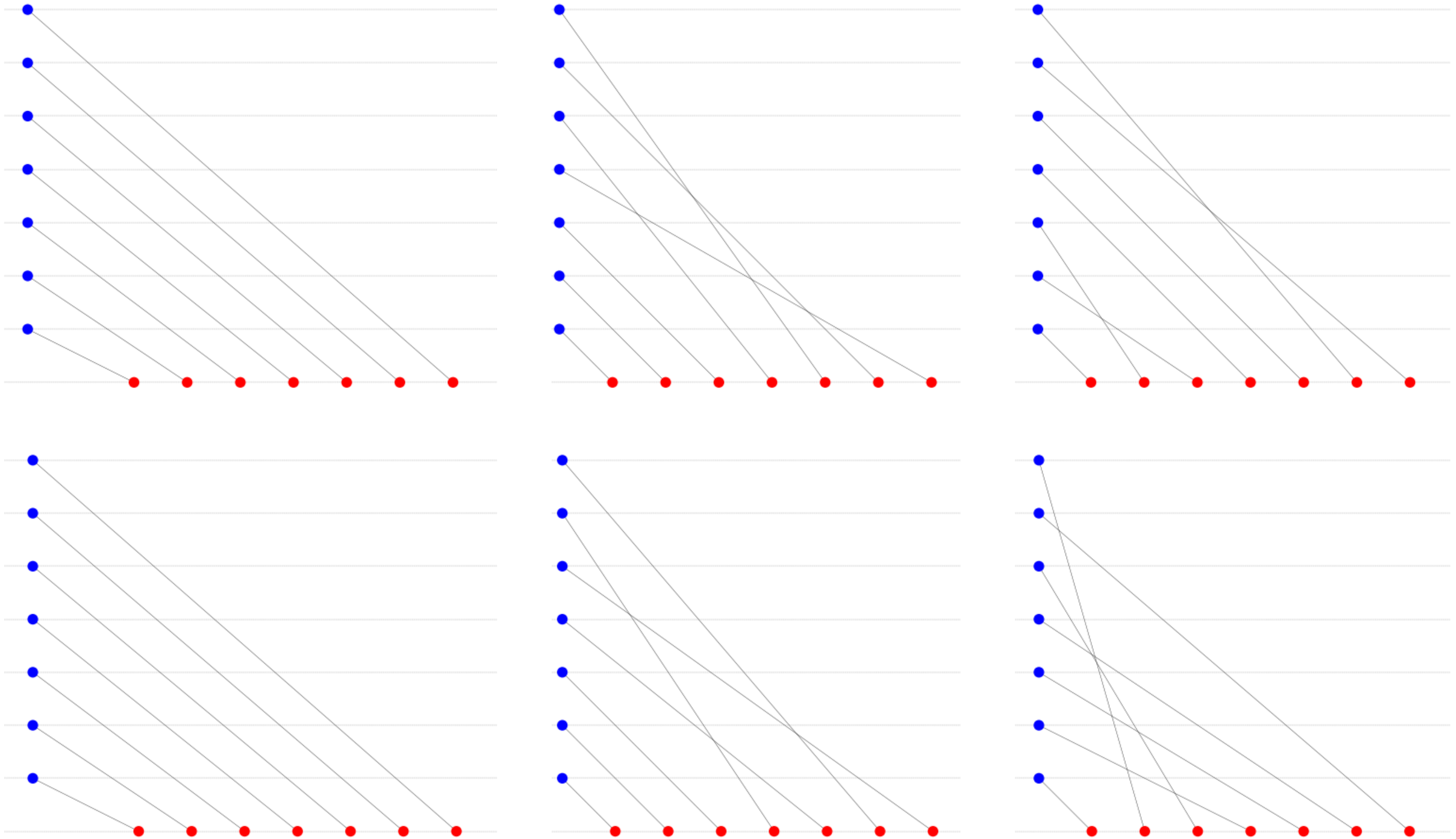
All receives guaranteed to occur
in same order run-to-run

Partially Non-Deterministic

Some receives guaranteed to
occur in same order

Run 1

Run 2



Deterministic

All receives guaranteed to occur in same order run-to-run

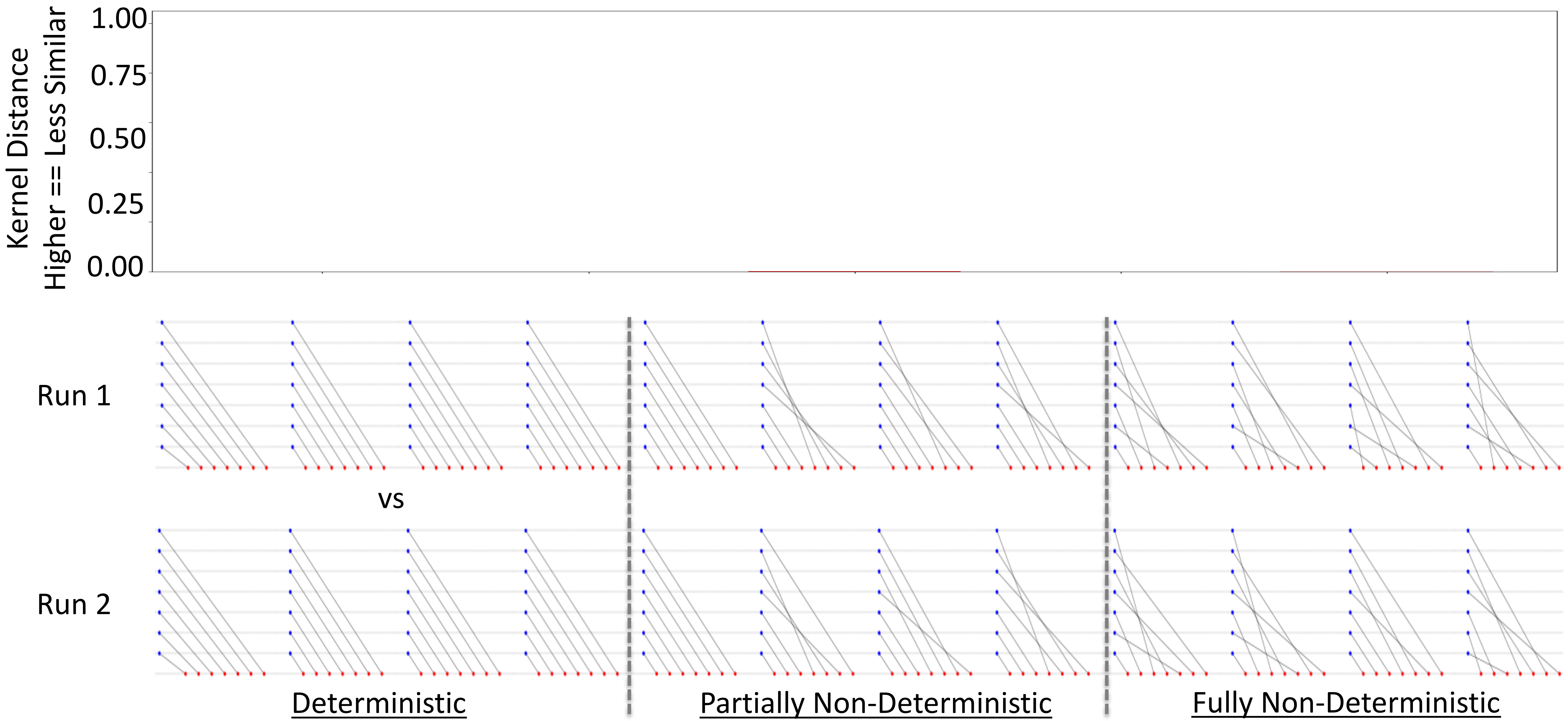
Partially Non-Deterministic

Some receives guaranteed to occur in same order

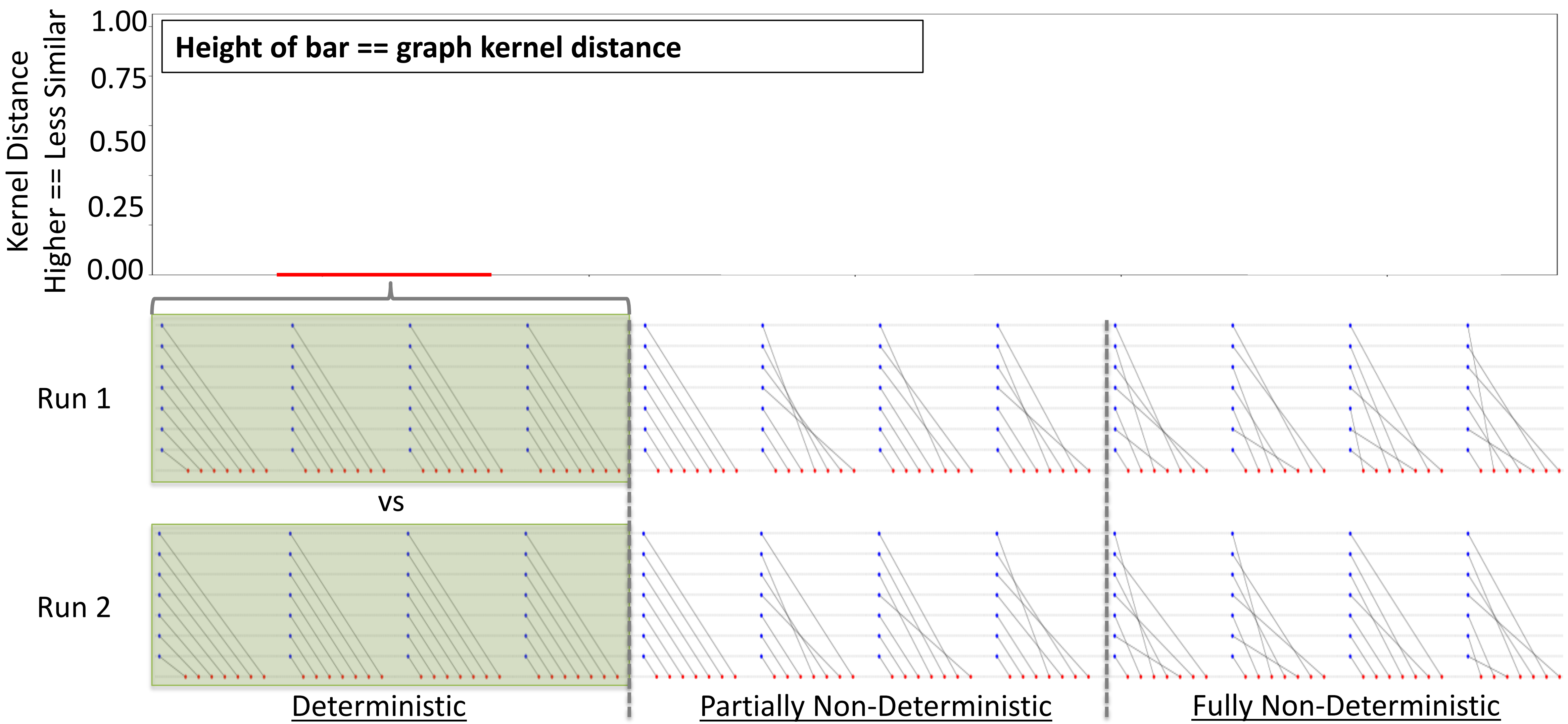
Fully Non-Deterministic

All receives occur in arbitrary order run-to-run

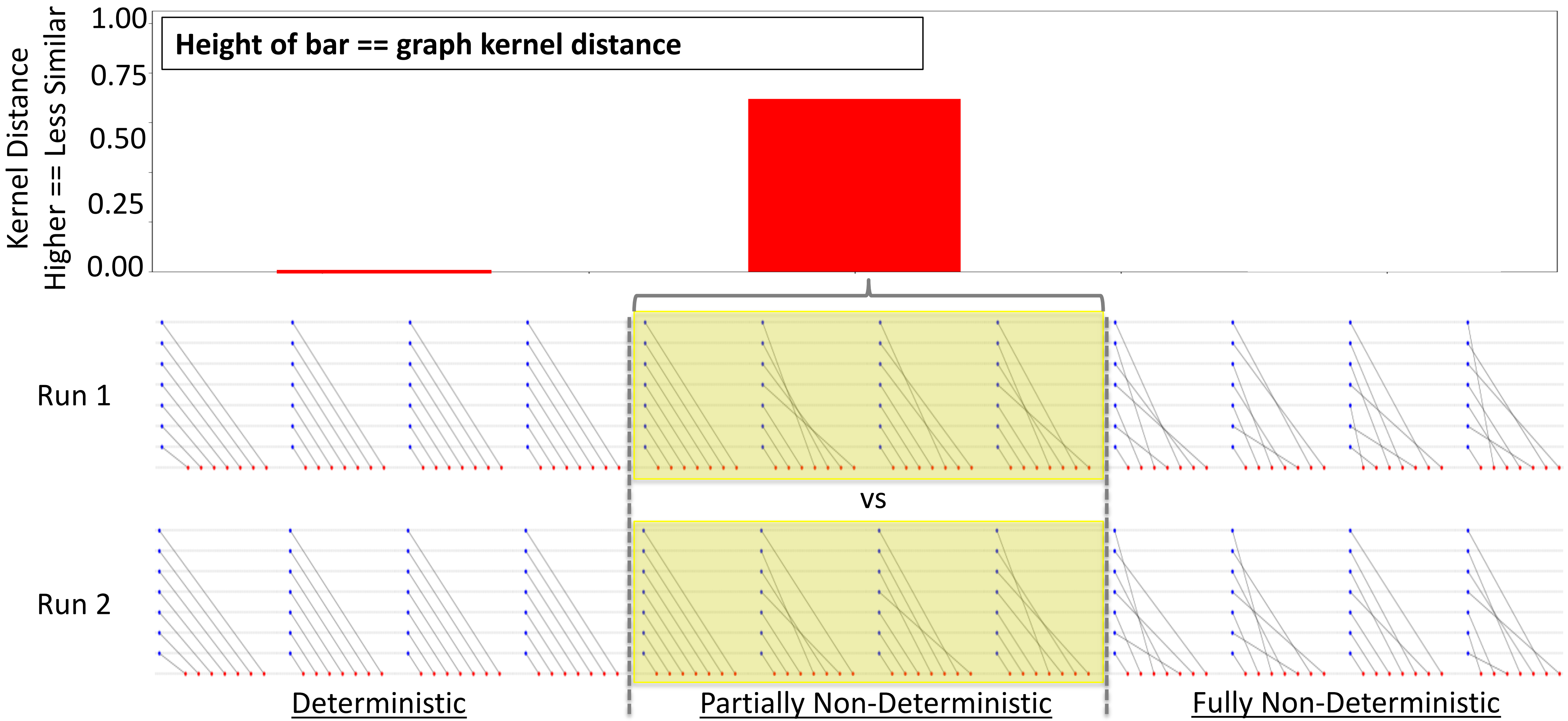
Comparing Runs via Kernel Distance



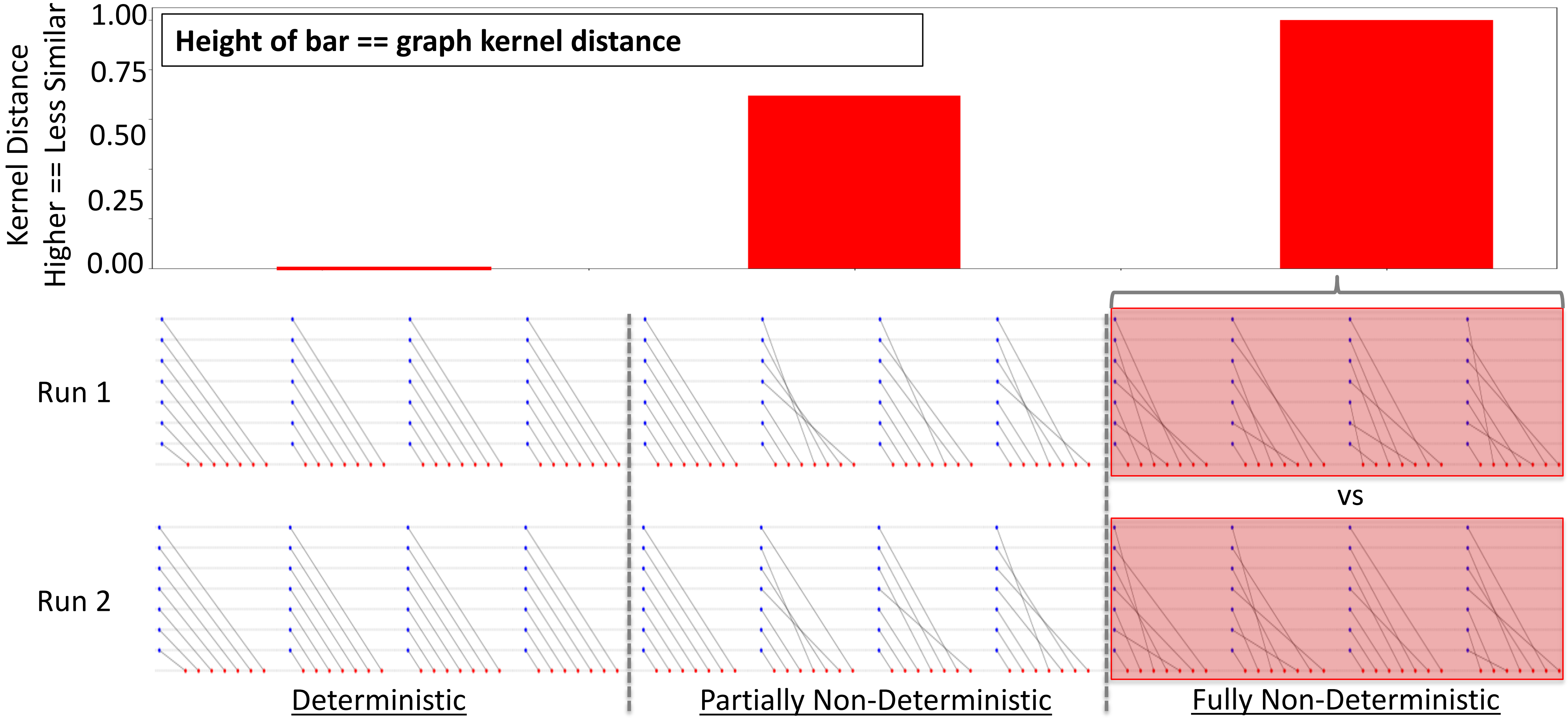
Comparing Runs via Kernel Distance



Comparing Runs via Kernel Distance



Comparing Runs via Kernel Distance



Kernel Distance Trends Across Multiple Runs

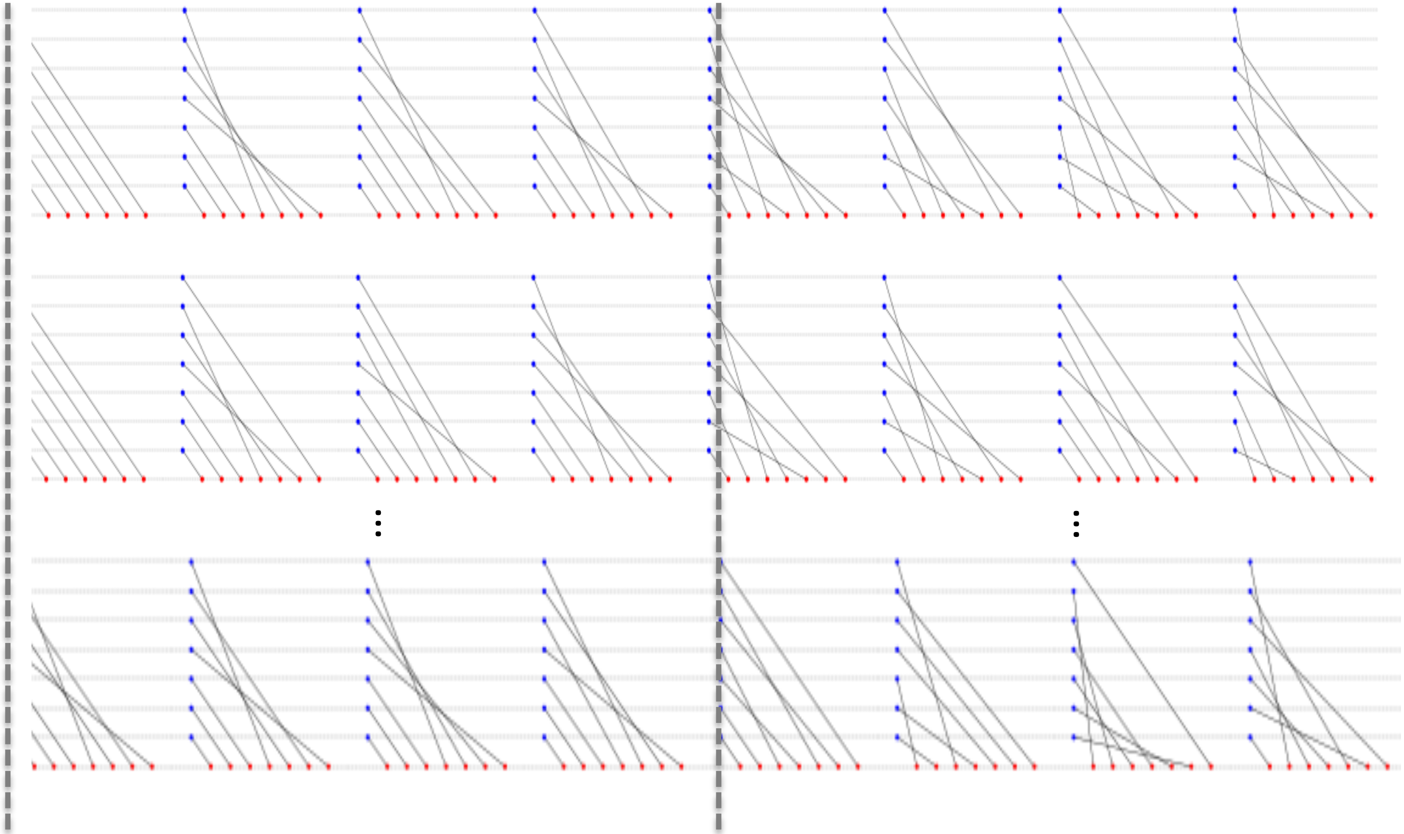
- Previous example only measured kernel distance between two executions
- But we want a statistical picture of the **trend in kernel distance over time across many executions**

Run 1

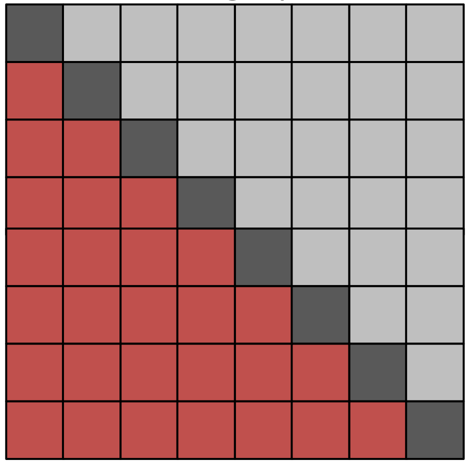
Run 2

⋮

Run N



Kernel matrix
w/r/t chosen graph kernel K

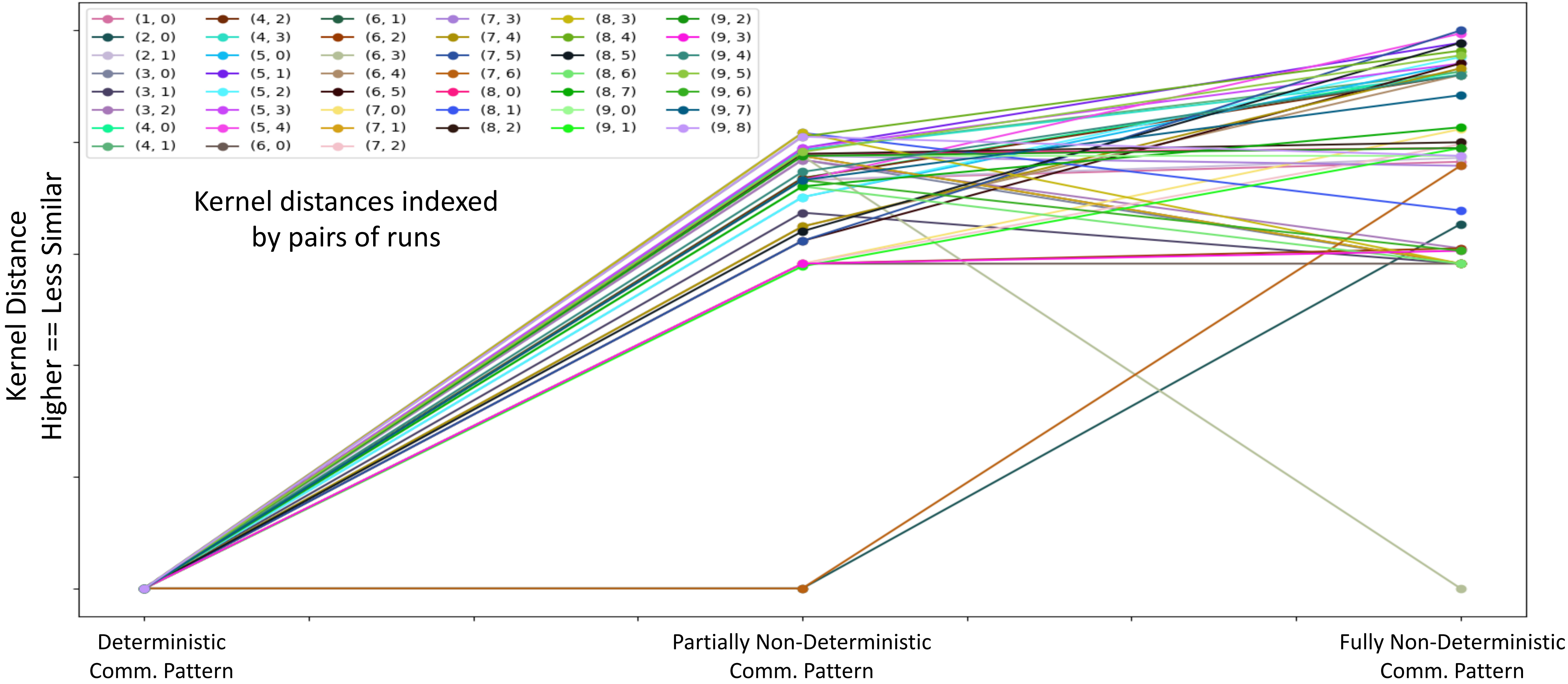


= kernel distance between graphs i and j

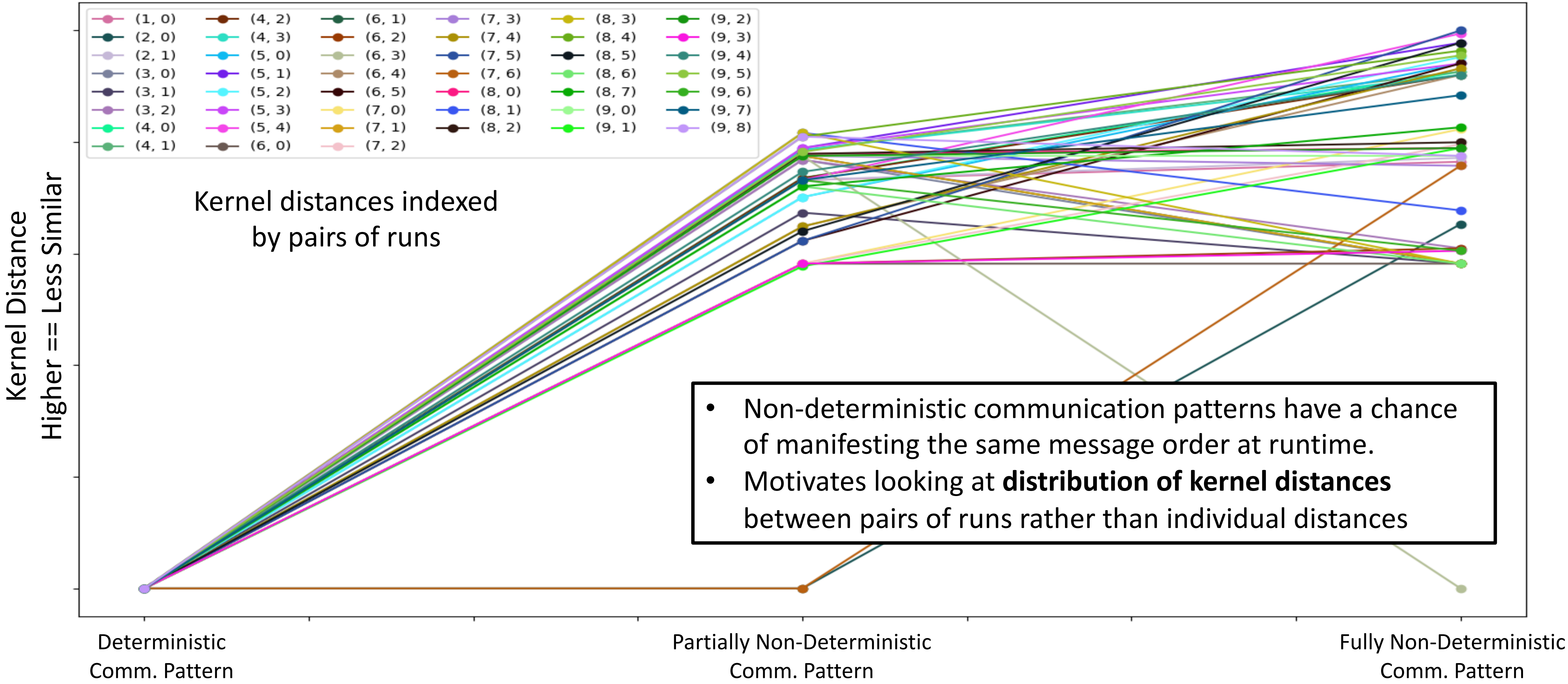
= kernel distance from graph to itself, always 0

= redundant, due to symmetry

Kernel Distance Trends Across Multiple Runs

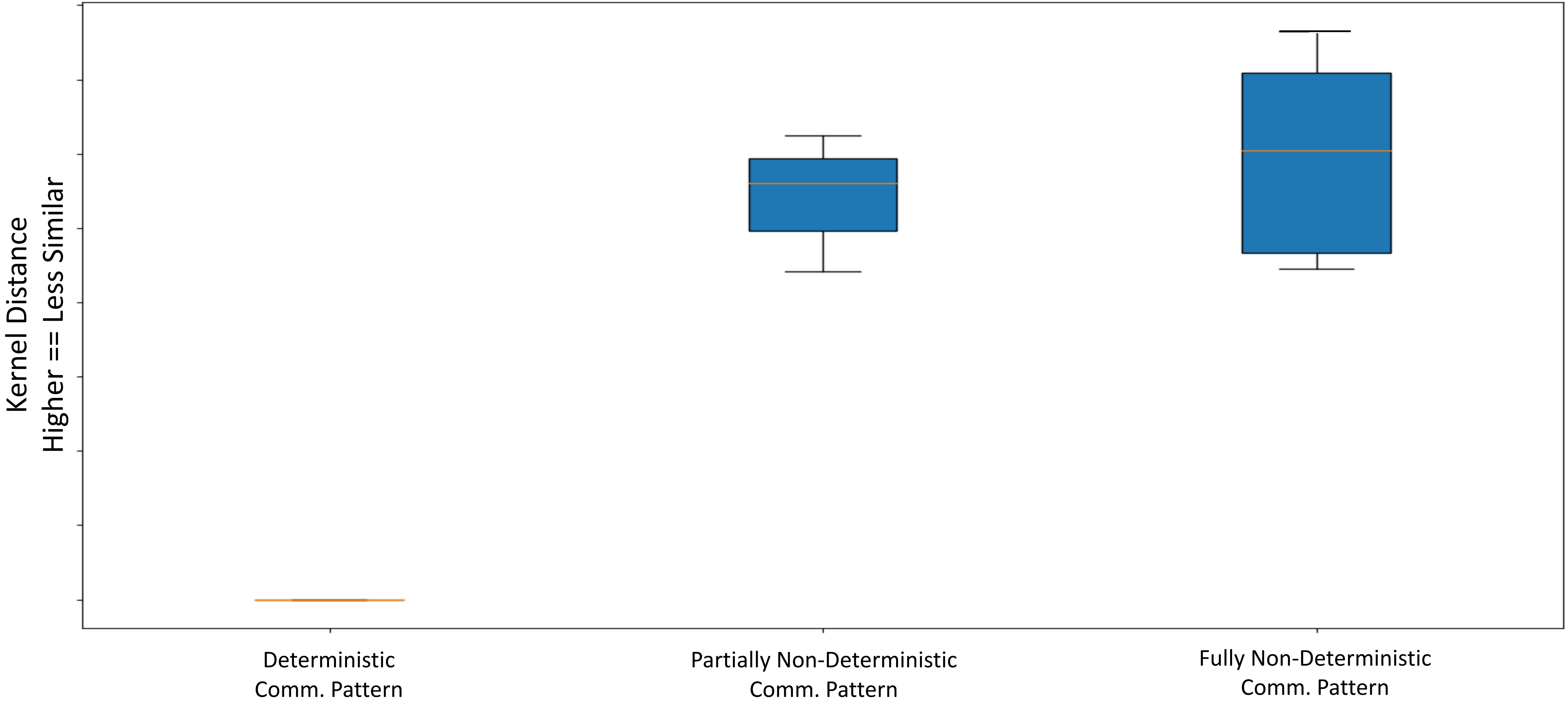


Kernel Distance Trends Across Multiple Runs



- Non-deterministic communication patterns have a chance of manifesting the same message order at runtime.
- Motivates looking at **distribution of kernel distances** between pairs of runs rather than individual distances

Kernel Distance Trends Across Multiple Runs



Evaluation Against a Realistic Communication Pattern

- We now evaluate against a communication pattern extracted from the AMG benchmark from the CORAL-2 Benchmark Suite

Evaluation Against a Realistic Communication Pattern

- We now evaluate against a communication pattern extracted from the AMG benchmark from the CORAL-2 Benchmark Suite
- Known to exhibit both receiver-side non-determinism and sender-side non-determinism [1]

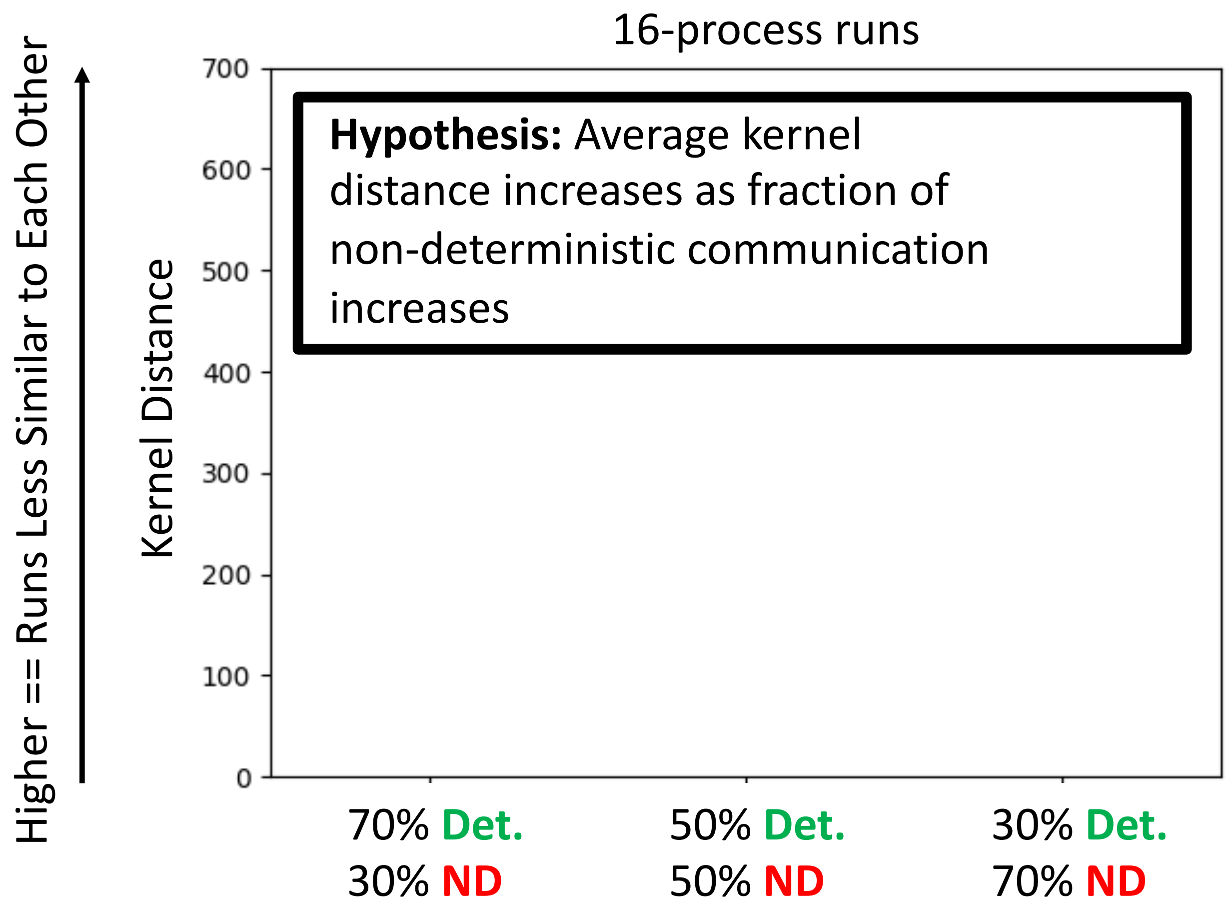
1. Cappello, F., Guermouche, A. and Snir, M., 2010, August. On communication determinism in parallel HPC applications. In *2010 Proceedings of 19th International Conference on Computer Communications and Networks* (pp. 1-8). IEEE.

Evaluation Against a Realistic Communication Pattern

- We now evaluate against a communication pattern extracted from the AMG benchmark from the CORAL-2 Benchmark Suite
- Known to exhibit both receiver-side non-determinism and sender-side non-determinism [1]
- For our evaluation, we control proportion of non-deterministic communication volume by splitting ranks into two groups:
  - One group performs the actual non-deterministic AMG pattern
  - One group performs a determinized version of the pattern

1. Cappello, F., Guermouche, A. and Snir, M., 2010, August. On communication determinism in parallel HPC applications. In *2010 Proceedings of 19th International Conference on Computer Communications and Networks* (pp. 1-8). IEEE.

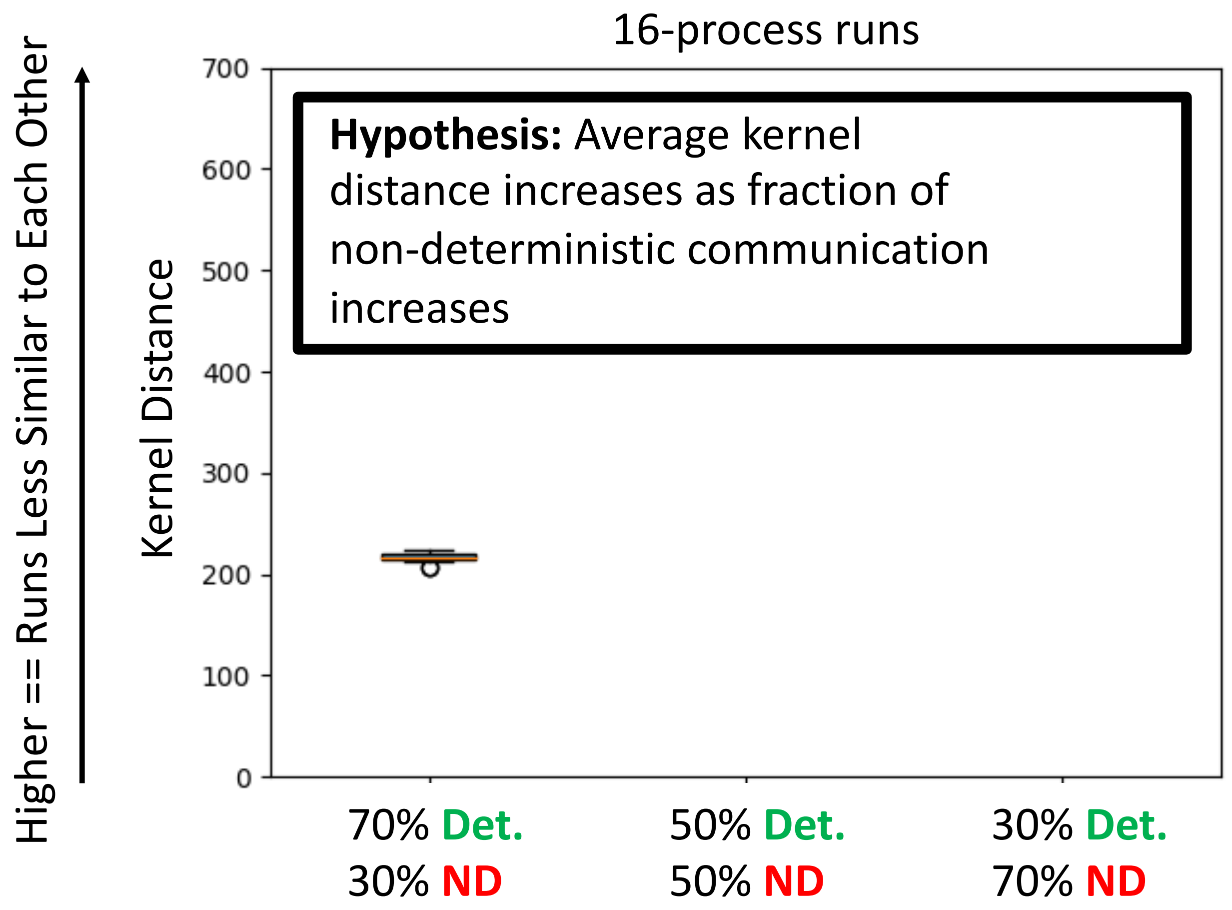
Kernel Distance Between Runs of Sequoia-AMG



Graph Kernel Used: Weisfeiler-Lehman Subtree Kernel [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.

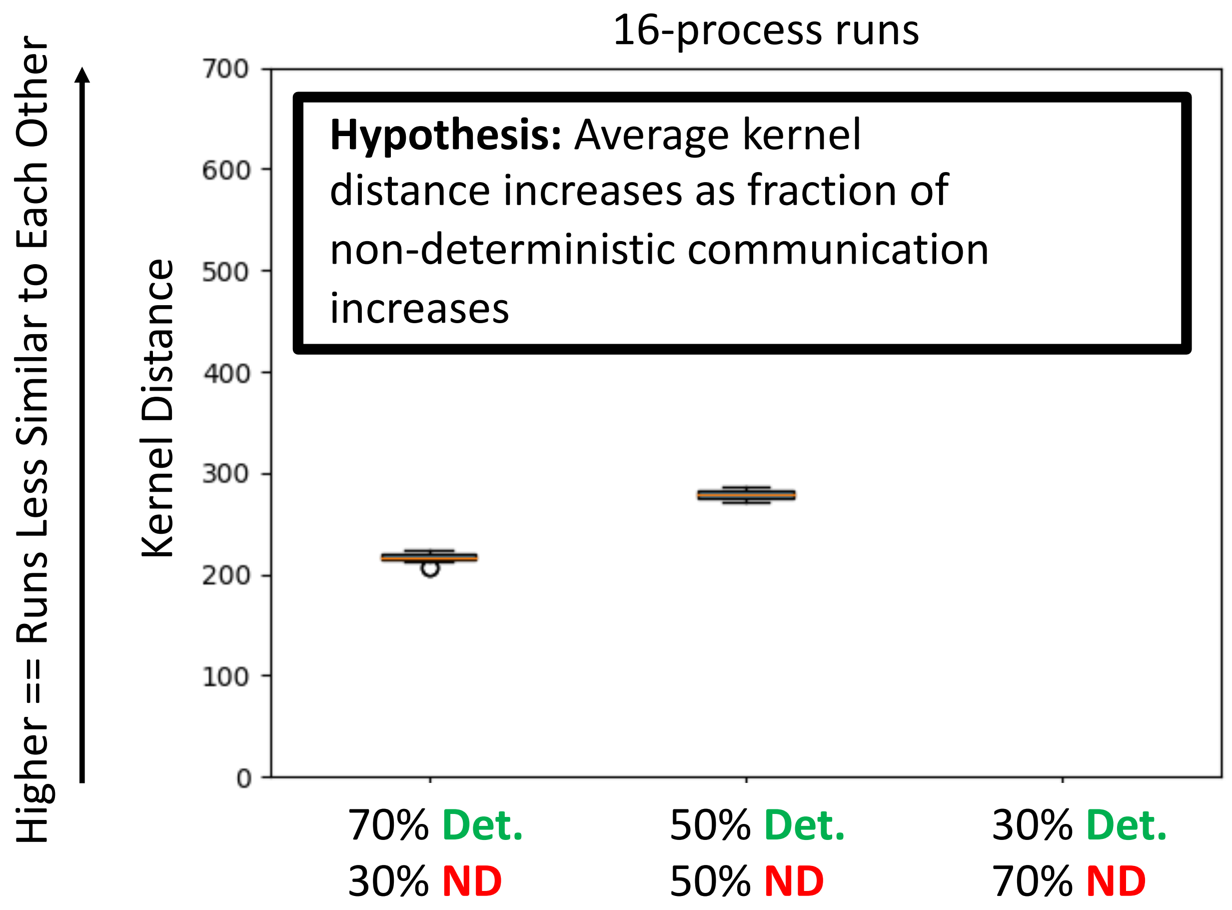
Kernel Distance Between Runs of Sequoia-AMG



Graph Kernel Used: Weisfeiler-Lehman Subtree Kernel [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.

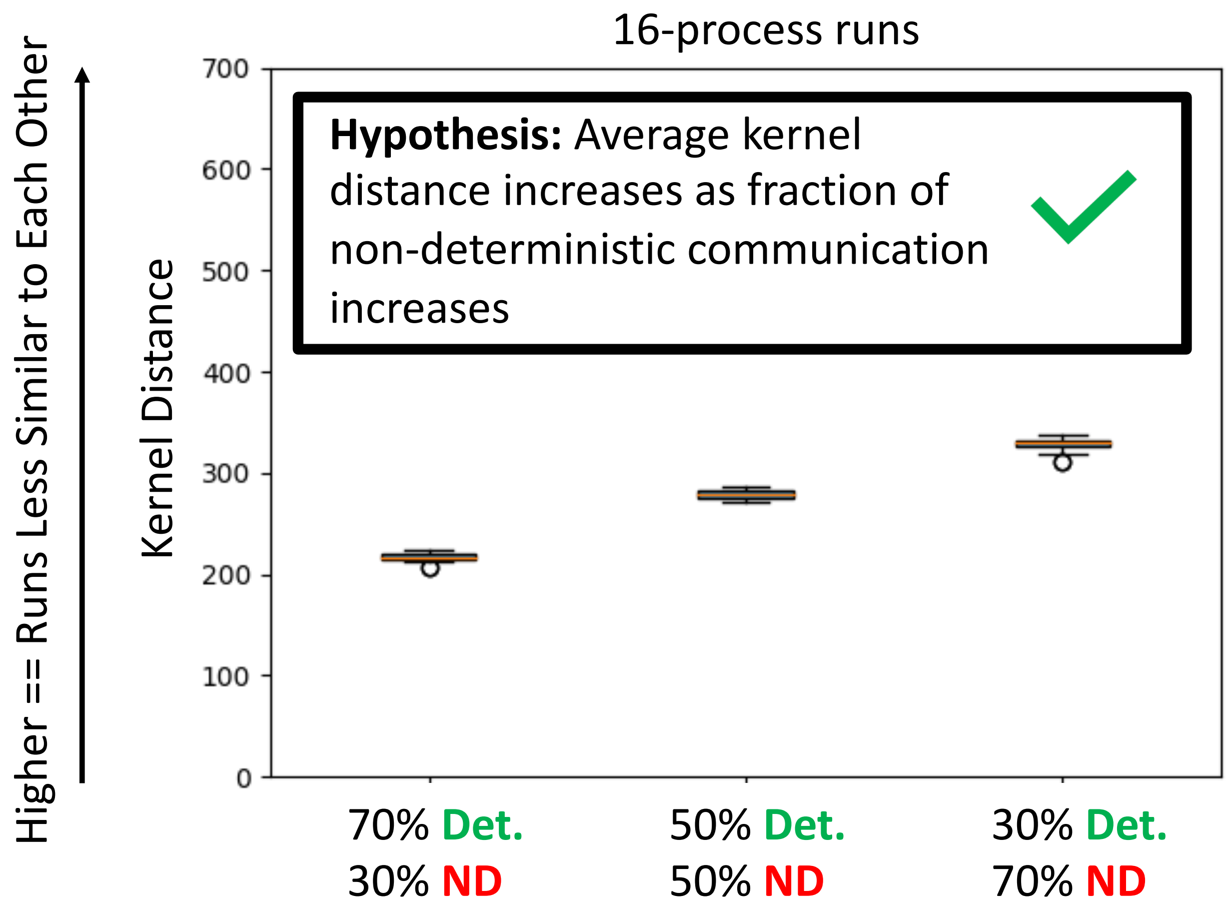
Kernel Distance Between Runs of Sequoia-AMG



Graph Kernel Used: Weisfeiler-Lehman Subtree Kernel [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.

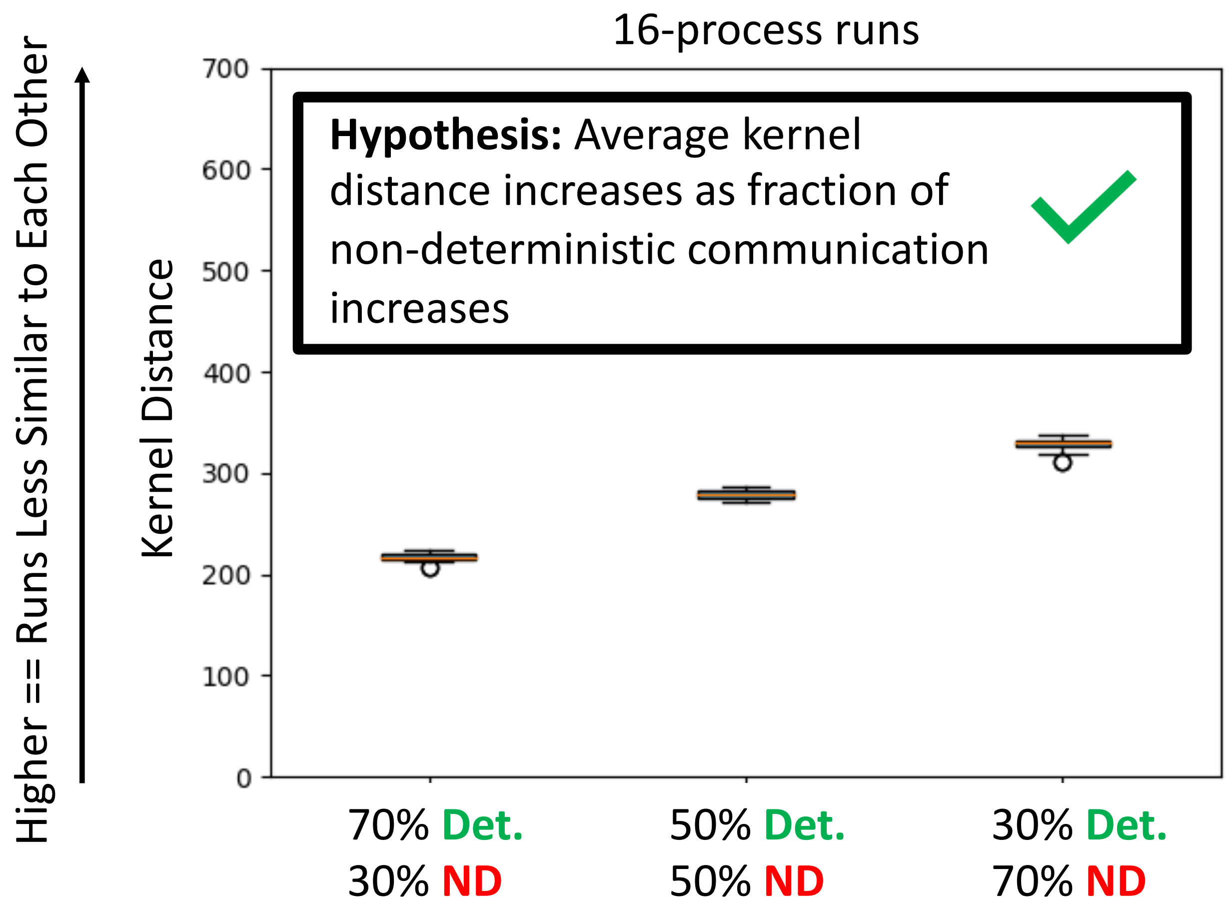
Kernel Distance Between Runs of Sequoia-AMG

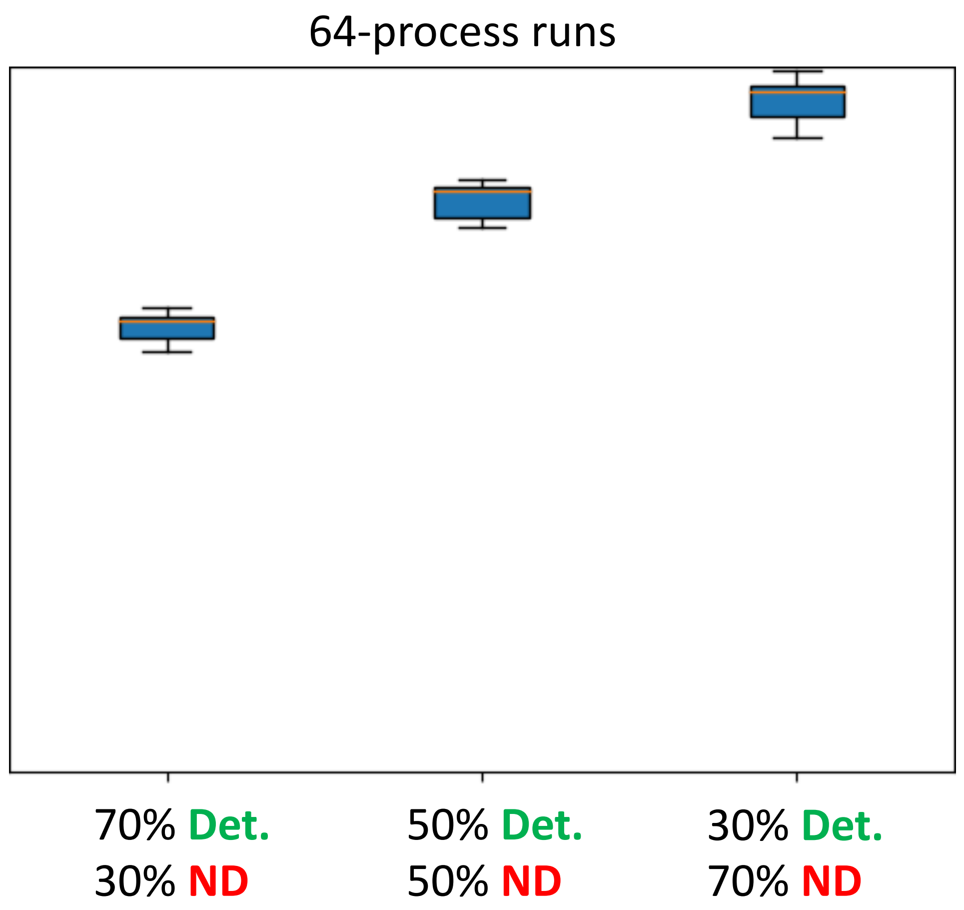


Graph Kernel Used: Weisfeiler-Lehman Subtree Kernel [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.

Kernel Distance Between Runs of Sequoia-AMG





Graph Kernel Used: Weisfeiler-Lehman Subtree Kernel [1]

1. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K. and Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), pp.2539-2561.

Lessons Learned

- Graph kernel distance is a useful proxy for degree of non-determinism in a communication pattern

Lessons Learned

- Graph kernel distance is a useful proxy for degree of non-determinism in a communication pattern
- This holds for both:
  - Simple communication patterns with only receiver-side non-determinism (e.g., naïve reduce pattern)
  - More complex patterns with mixed receiver/sender-side non-determinism (e.g., Sequoia-AMG Iprobe pattern)

Workflow for Non-Determinism Characterization

- **Phase 1:** Build graph-structured models of executions
- **Phase 2:** Quantify cross-execution trends in non-deterministic communication via graph similarity
- **Phase 3**: Detect periods of anomalous execution dissimilarity and localize potential root causes

Applying our Workflow to miniAMR

- Adaptive mesh refinement (AMR) code from the Matevo Benchmark Suite [1]

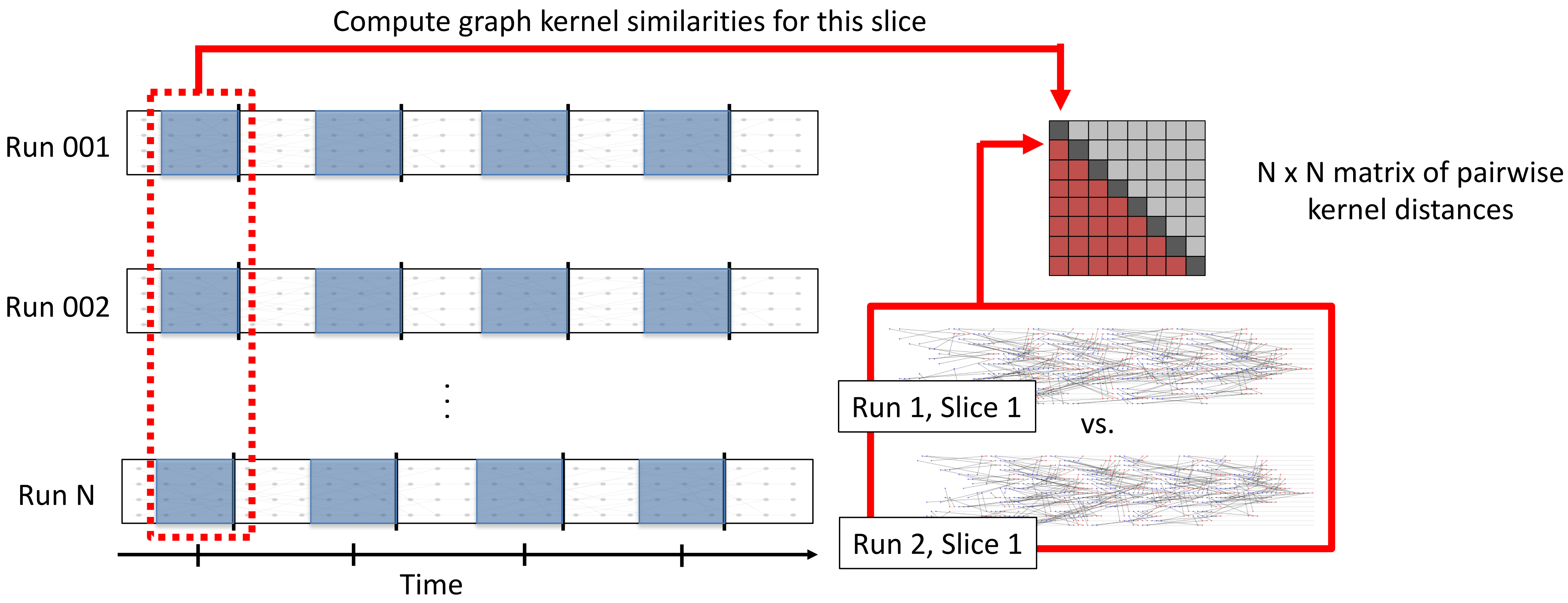
1. Heroux, M.A., Doerfler, D.W., Crozier, P.S., Willenbring, J.M., Edwards, H.C., Williams, A., Rajan, M., Keiter, E.R., Thornquist, H.K. and Numrich, R.W., 2009. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, *3*.

Applying our Workflow to miniAMR

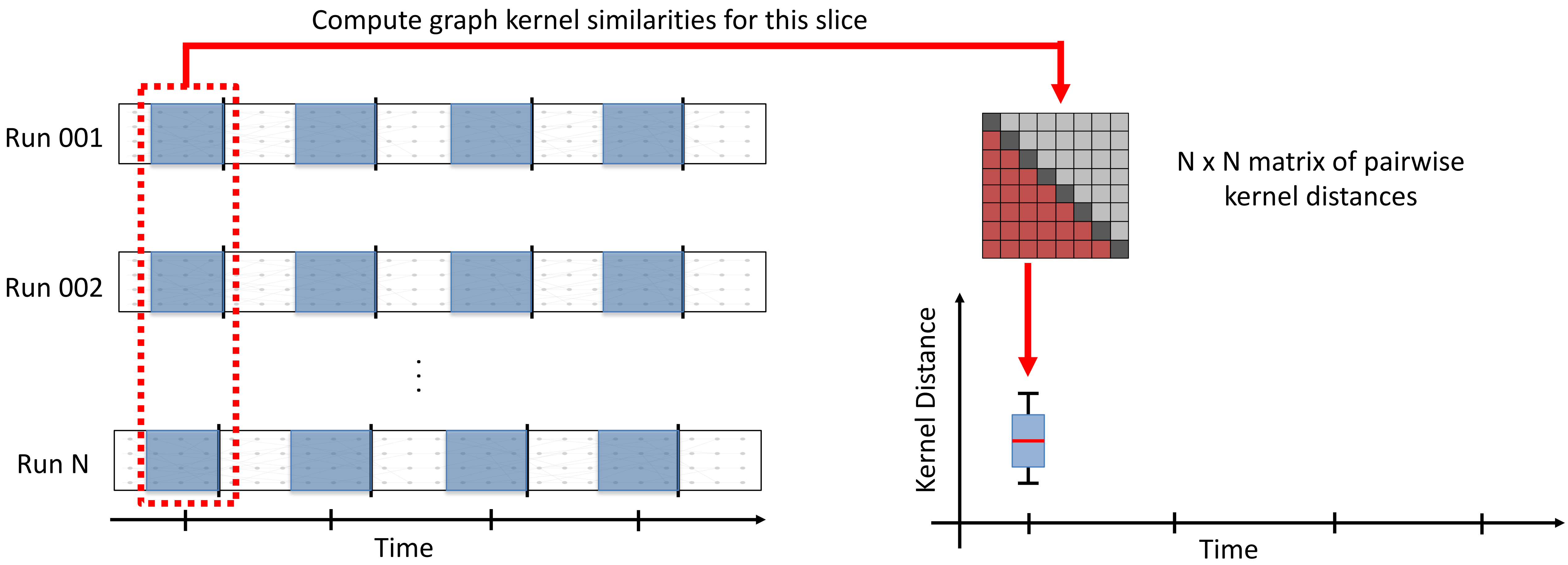
- Adaptive mesh refinement (AMR) code from the Matevo Benchmark Suite [1]
- We target miniAMR based on the following criteria:
  - MPI application exhibiting communication non-determinism
  - Root cause of non-determinism known *a priori*

1. Heroux, M.A., Doerfler, D.W., Crozier, P.S., Willenbring, J.M., Edwards, H.C., Williams, A., Rajan, M., Keiter, E.R., Thornquist, H.K. and Numrich, R.W., 2009. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, *3*.

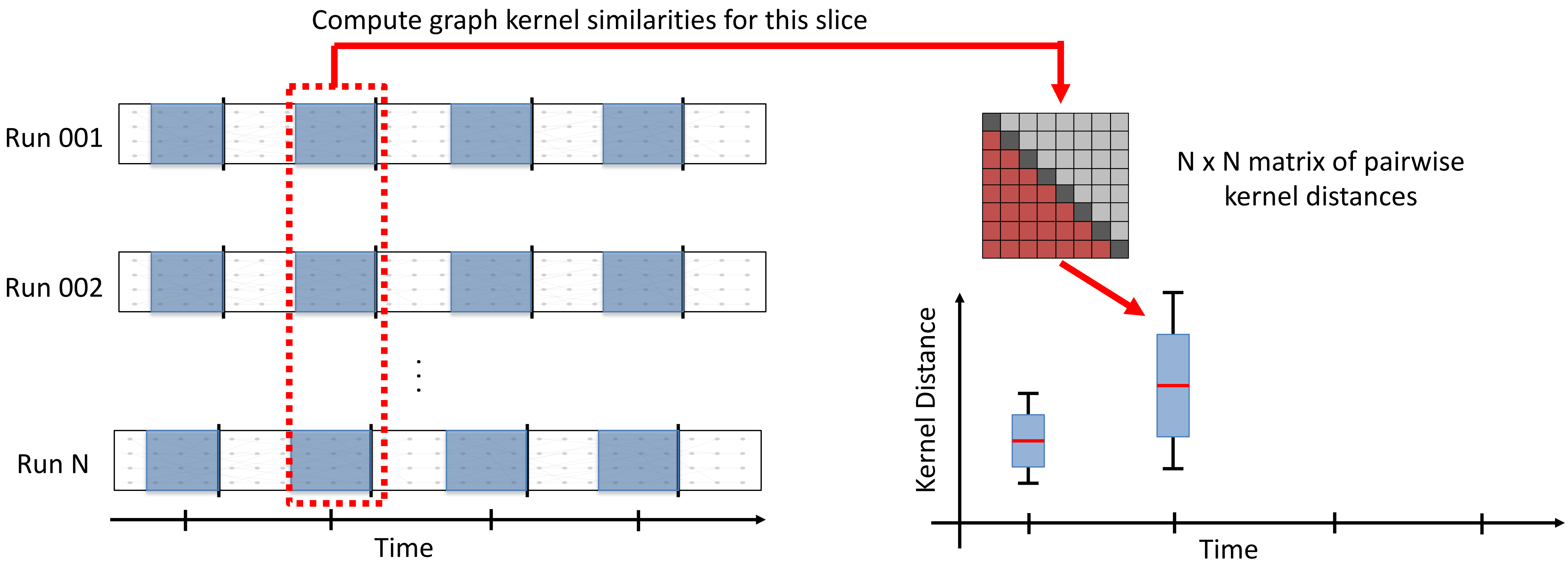
Event Graph Slicing



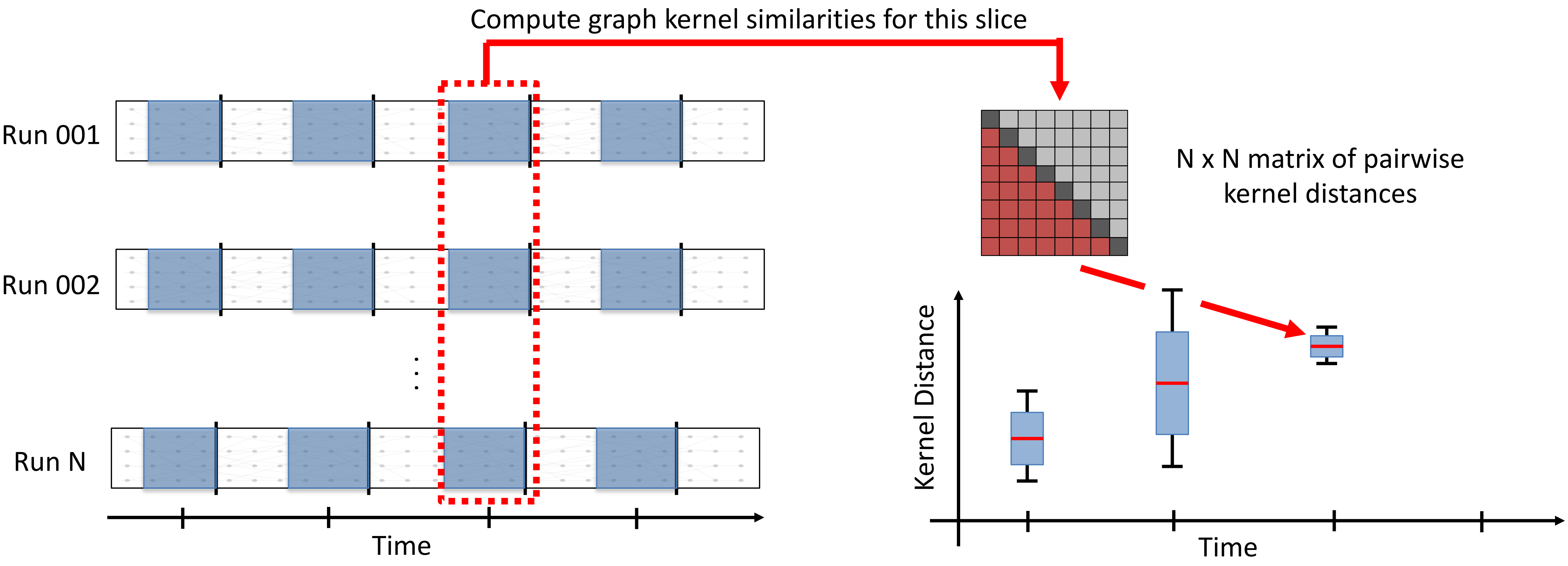
Distributions of Kernel Distances



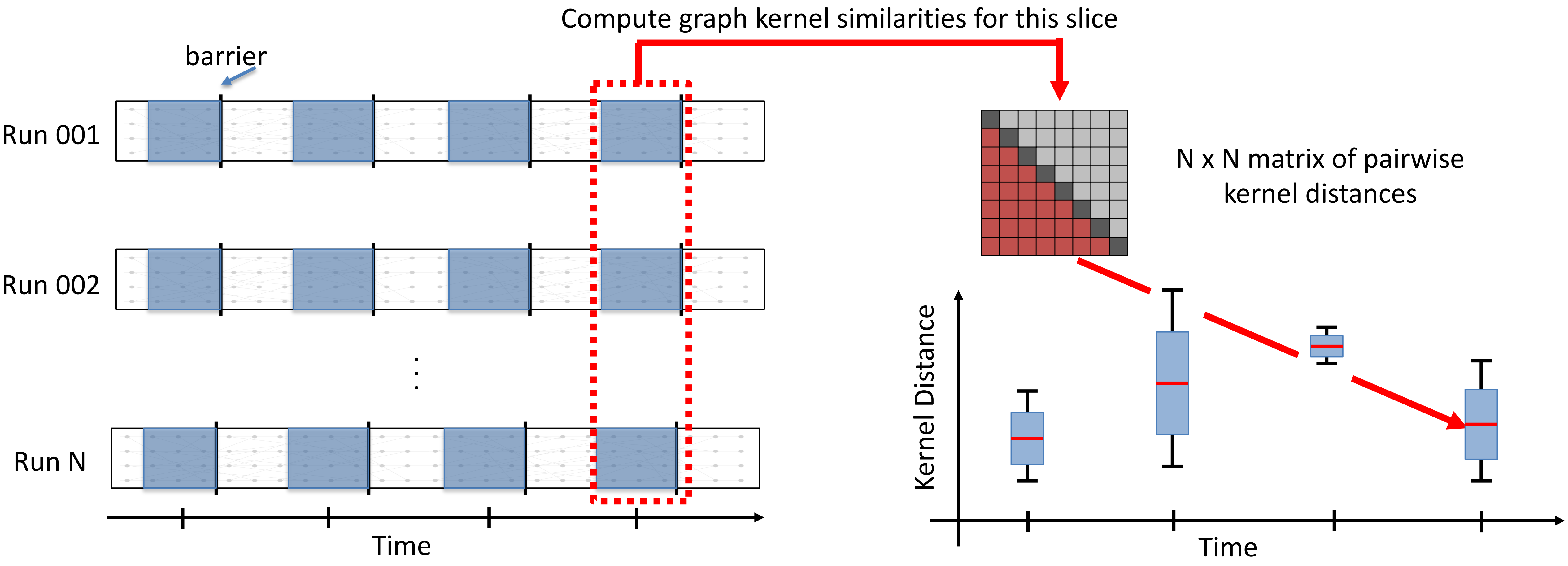
Kernel Distance Trends Over Time



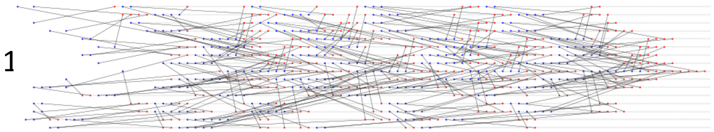
Kernel Distance Trends Over Time

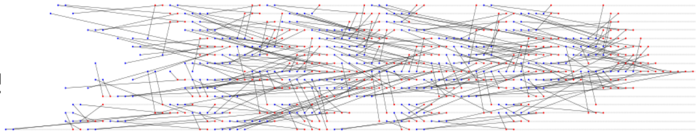


Kernel Distance Trends Over Time

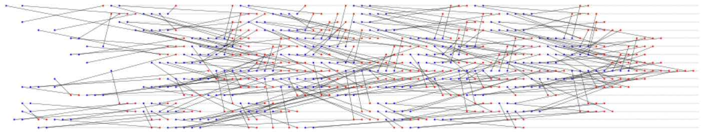


Linking Observed Non-Determinism to Root Causes

Run 1


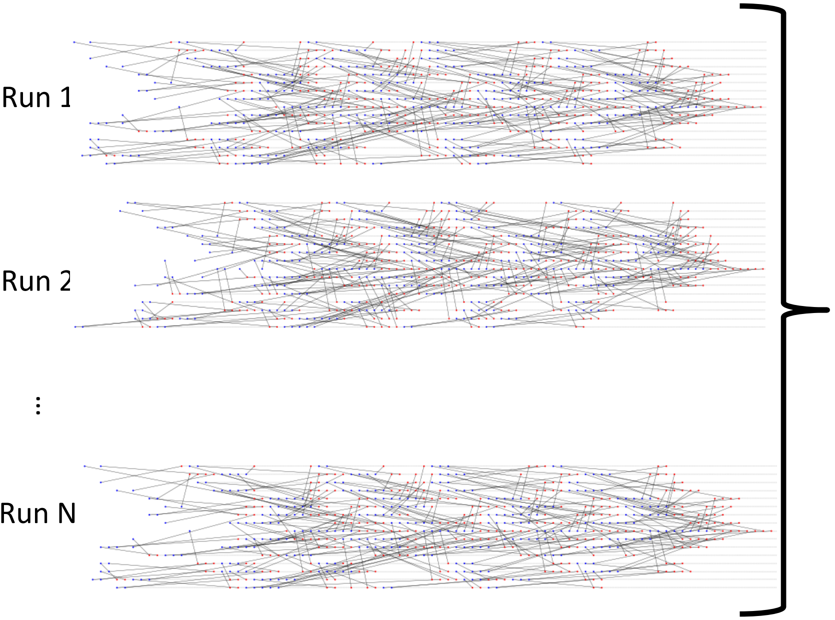
Run 2


⋮

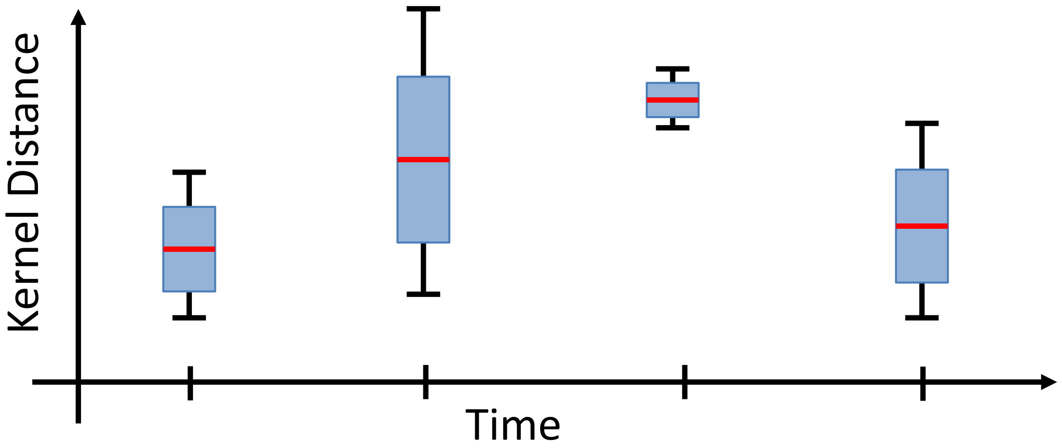
Run N


A set of N event graphs
modeling N runs of a
non-deterministic application

Linking Observed Non-Determinism to Root Causes

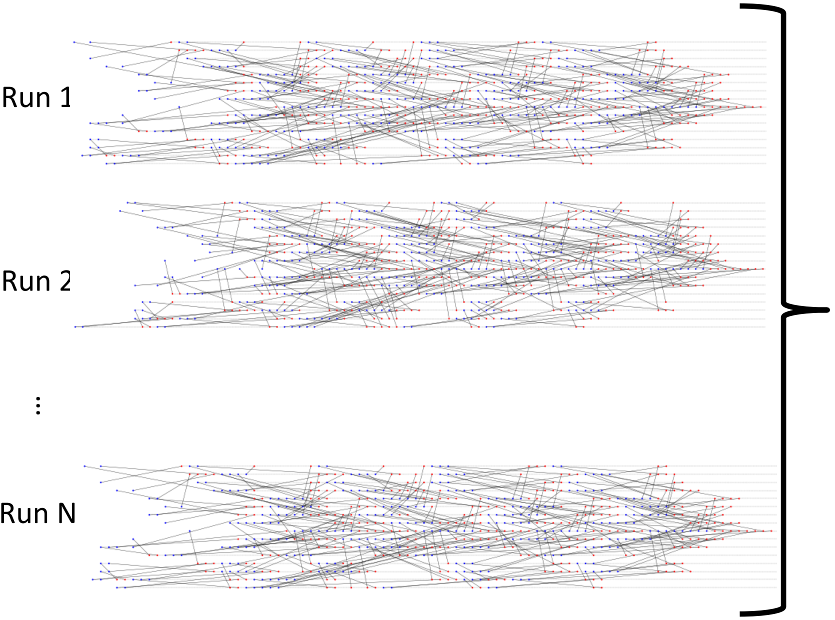


A set of N event graphs
modeling N runs of a
non-deterministic application

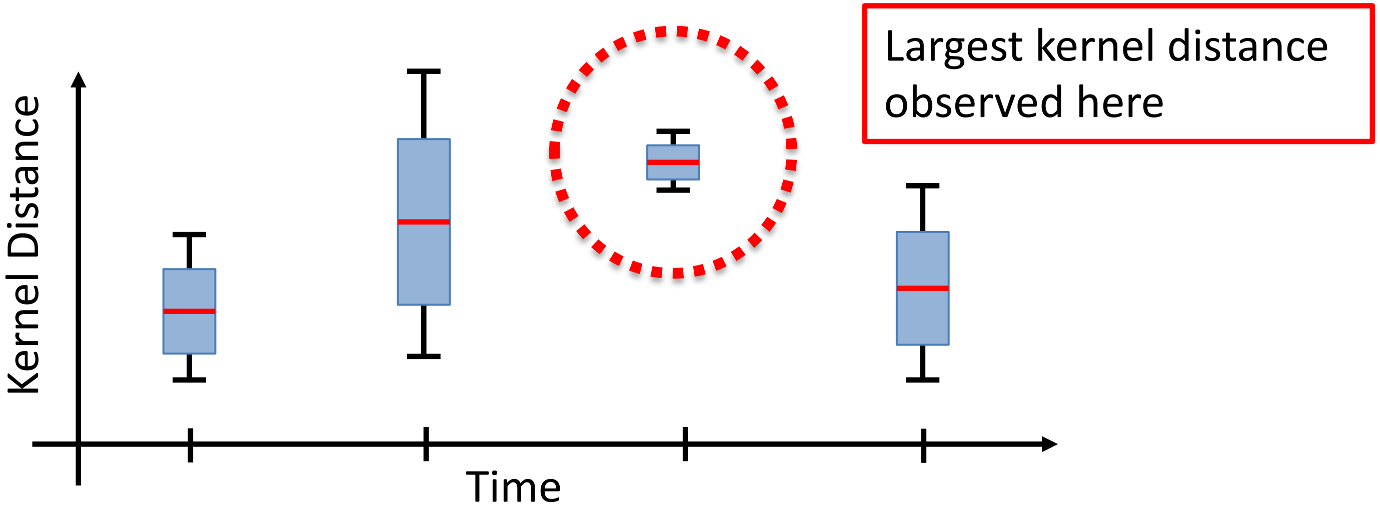


A time-series of distributions of graph kernel distances
between corresponding subgraphs
from pairs of runs

Linking Observed Non-Determinism to Root Causes

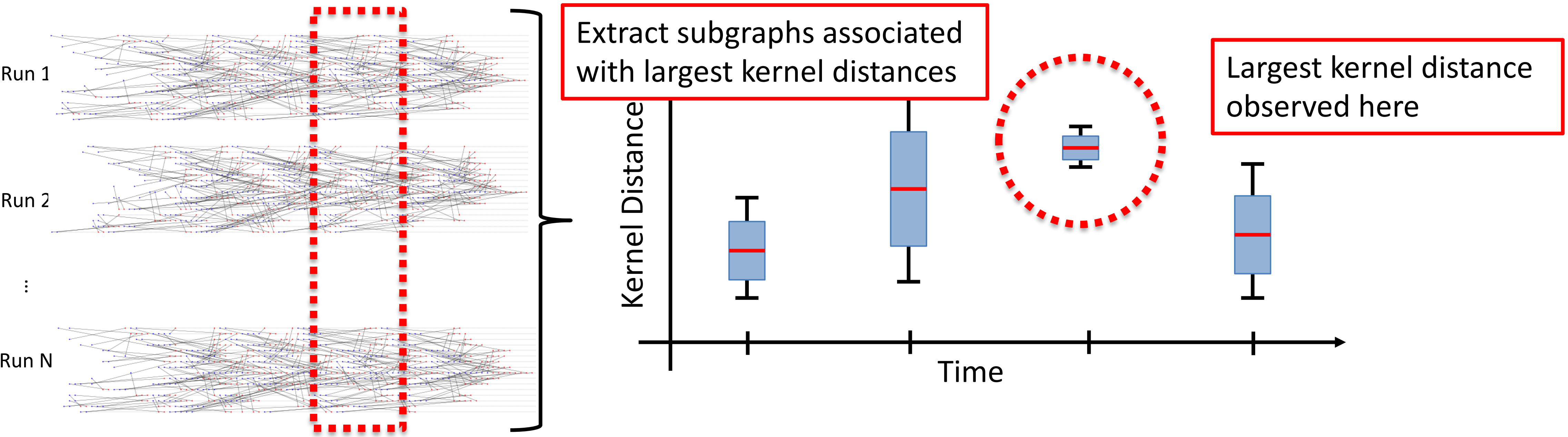


A set of N event graphs
modeling N runs of a
non-deterministic application



A time-series of distributions of graph kernel distances
between corresponding subgraphs
from pairs of runs

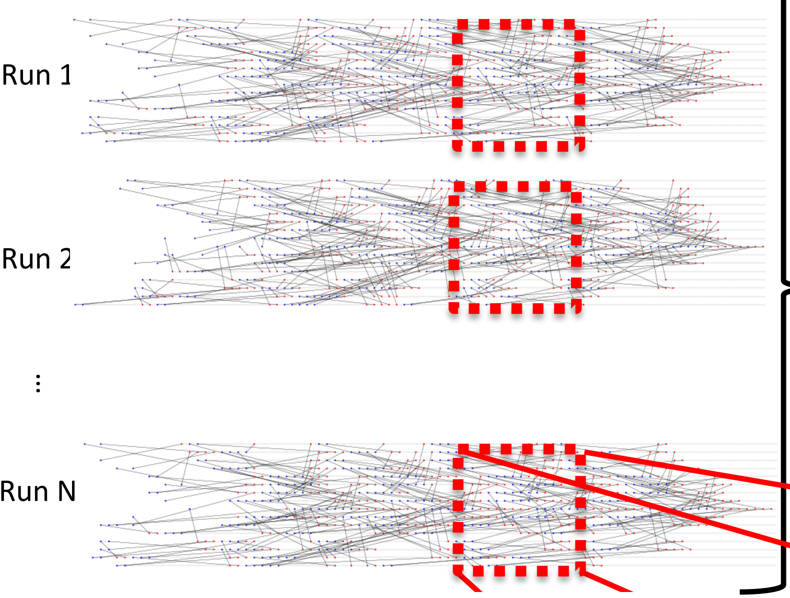
Linking Observed Non-Determinism to Root Causes



A set of N event graphs modeling N runs of a non-deterministic application

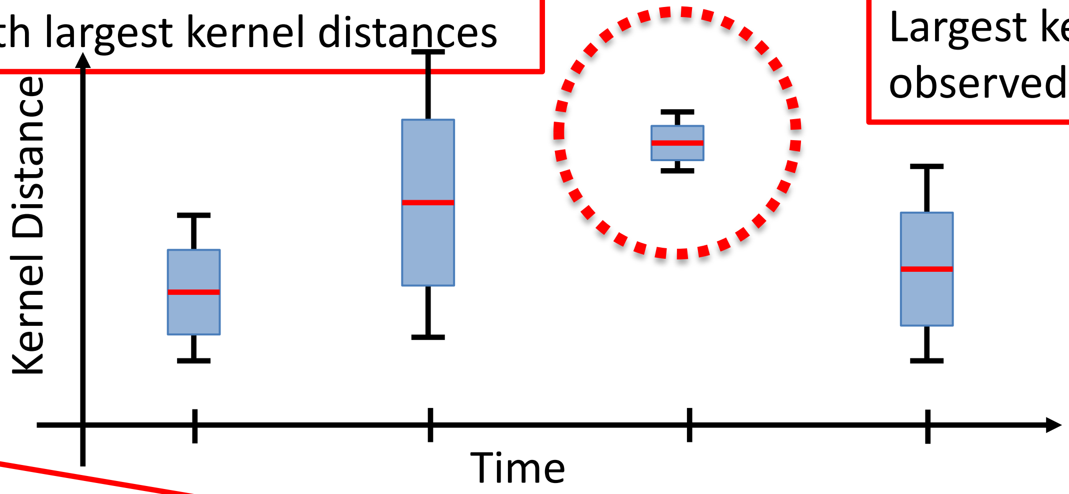
A time-series of distributions of graph kernel distances between corresponding subgraphs from pairs of runs

Linking Observed Non-Determinism to Root Causes



A set of N event graphs modeling N runs of a non-deterministic application

Extract subgraphs associated with largest kernel distances

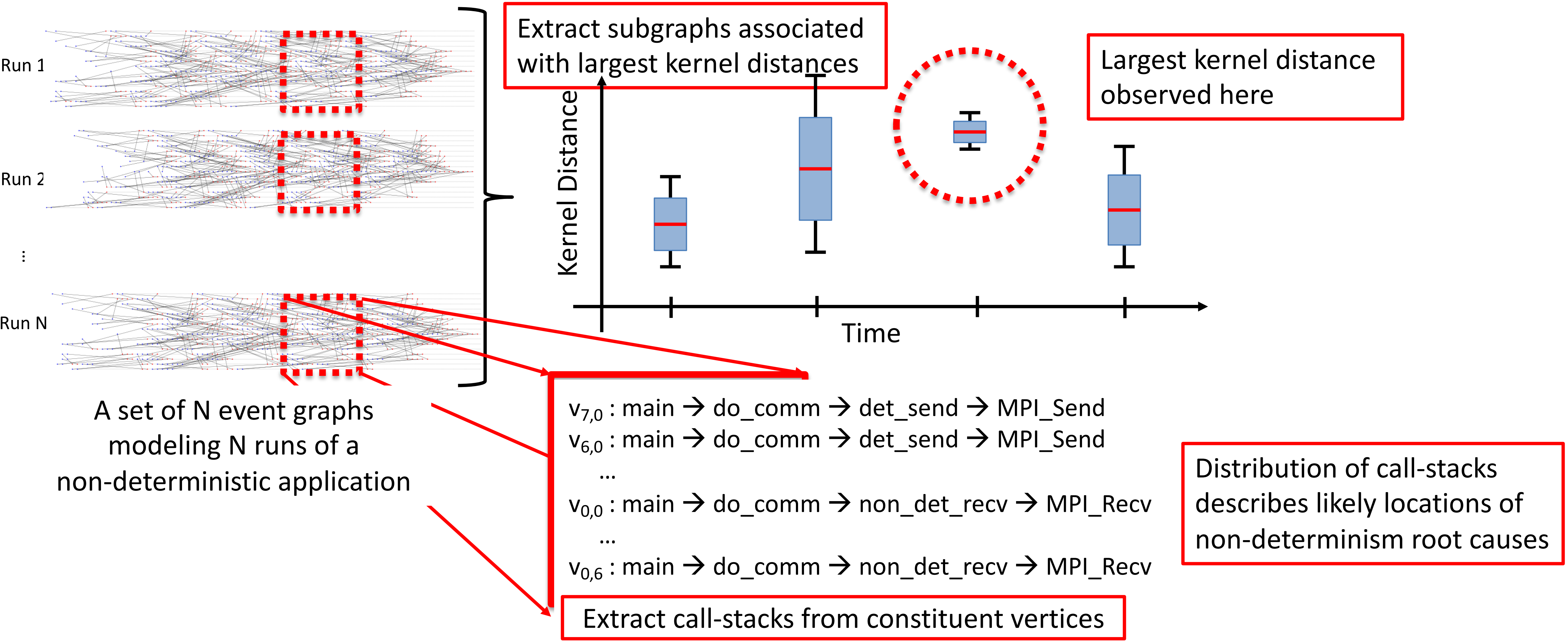


Largest kernel distance observed here

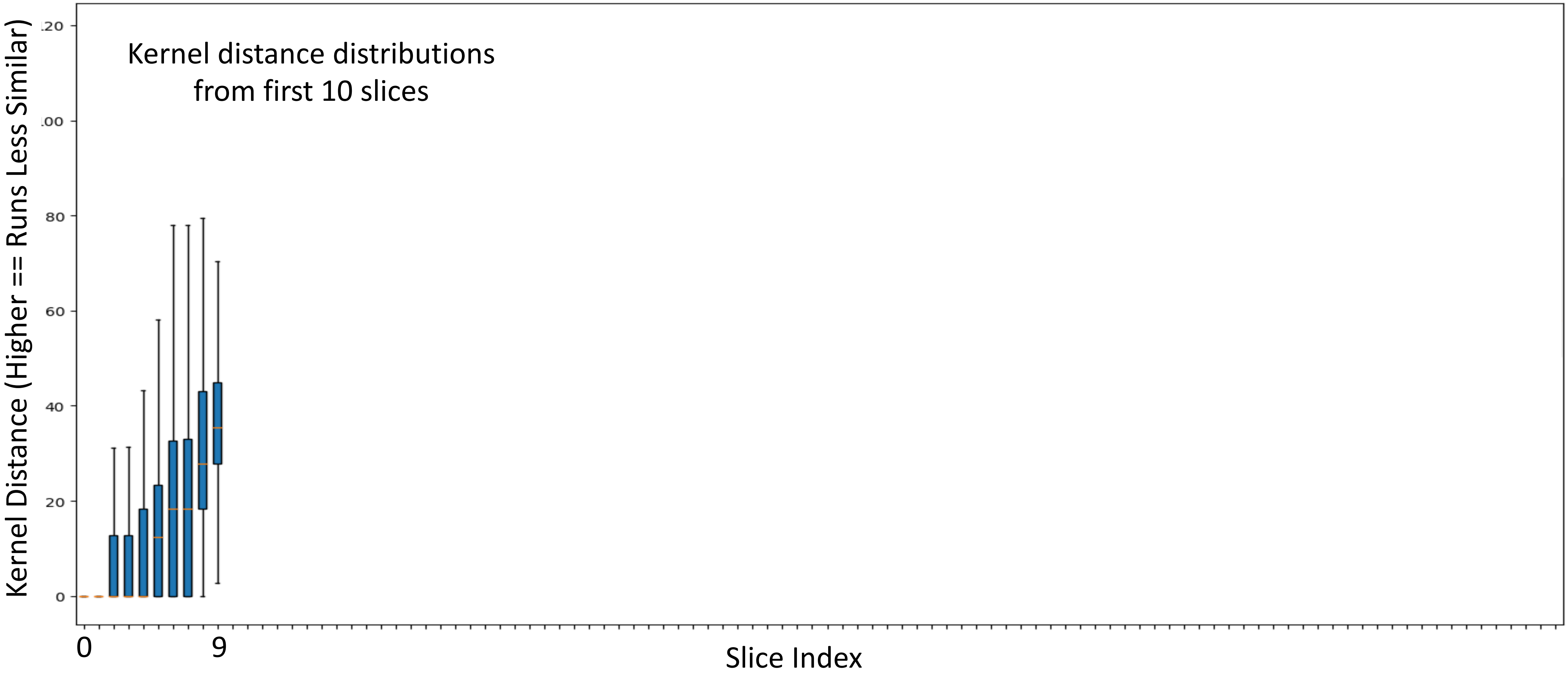
$v_{7,0}$: main → do_comm → det_send → MPI_Send
$v_{6,0}$: main → do_comm → det_send → MPI_Send
…
$v_{0,0}$: main → do_comm → non_det_recv → MPI_Recv
…
$v_{0,6}$: main → do_comm → non_det_recv → MPI_Recv

Extract call-stacks from constituent vertices

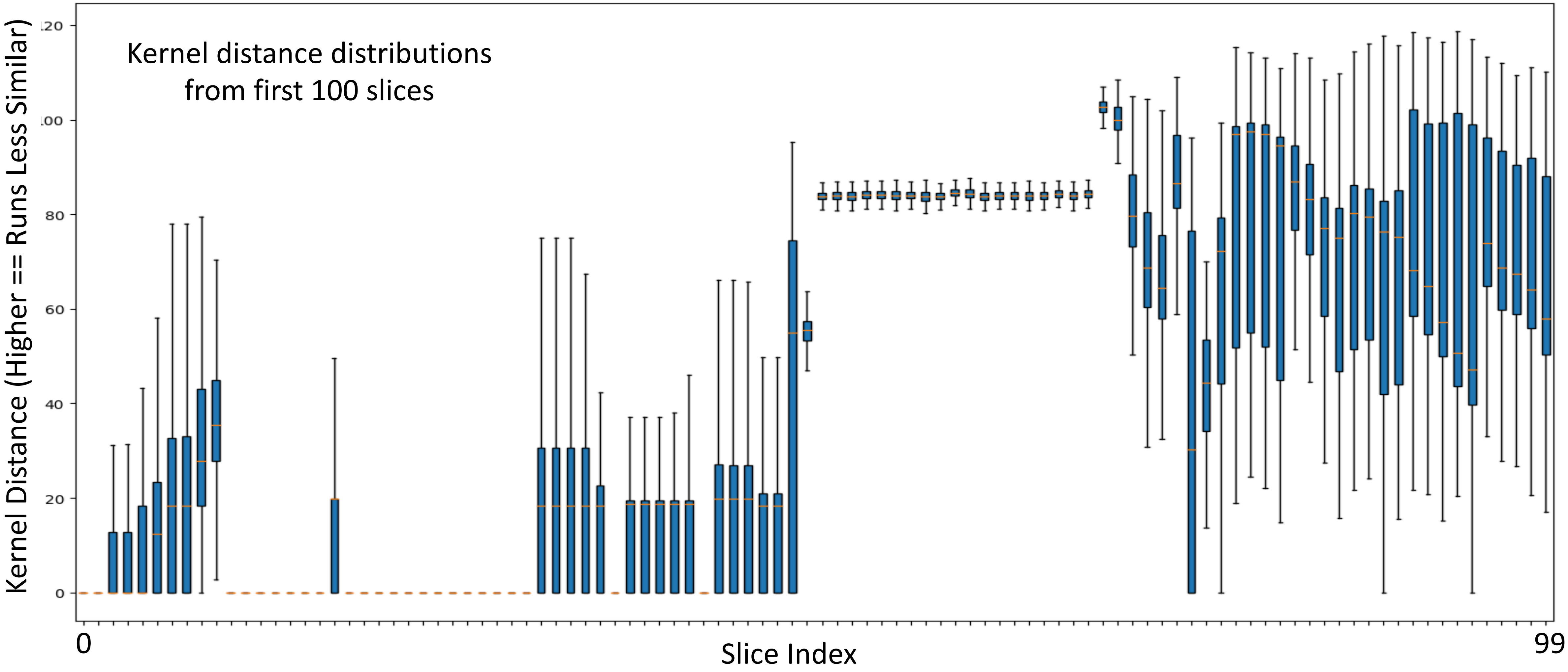
Linking Observed Non-Determinism to Root Causes



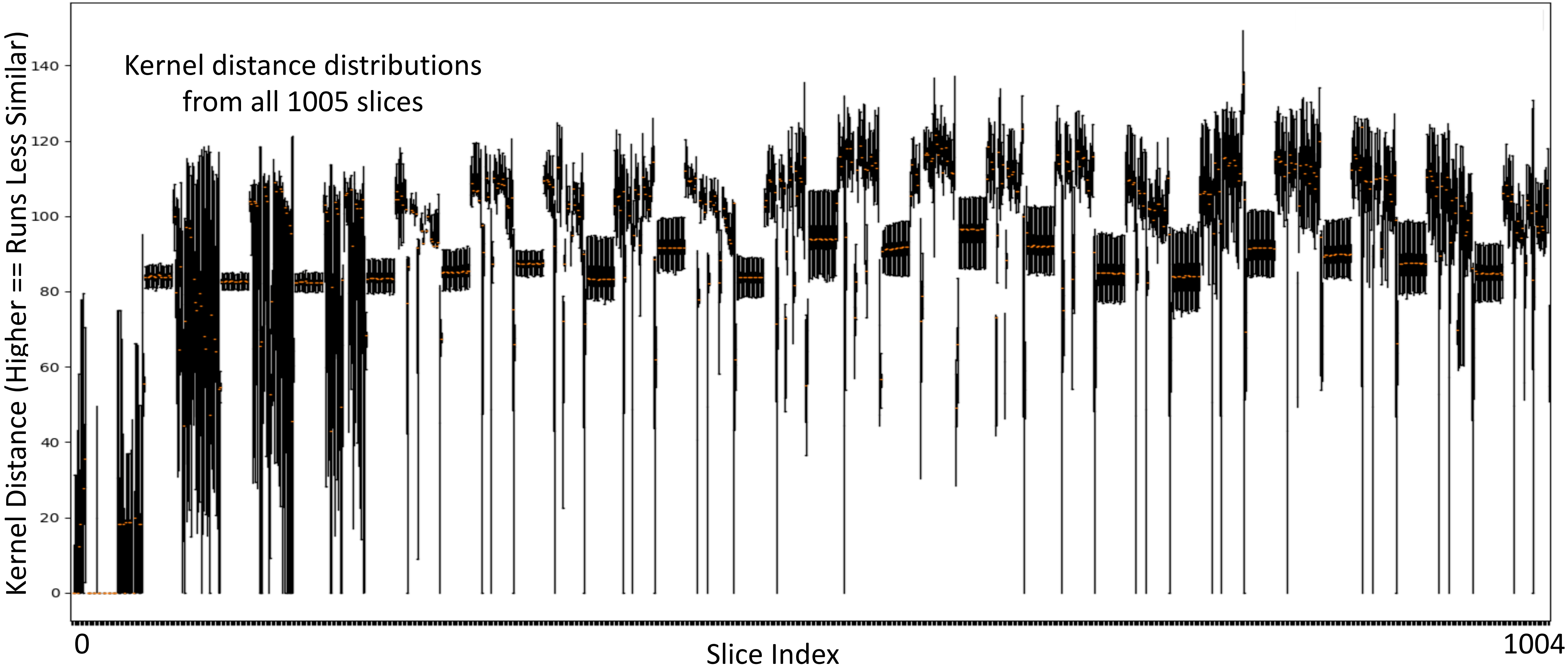
Kernel Distance Time Series for miniAMR



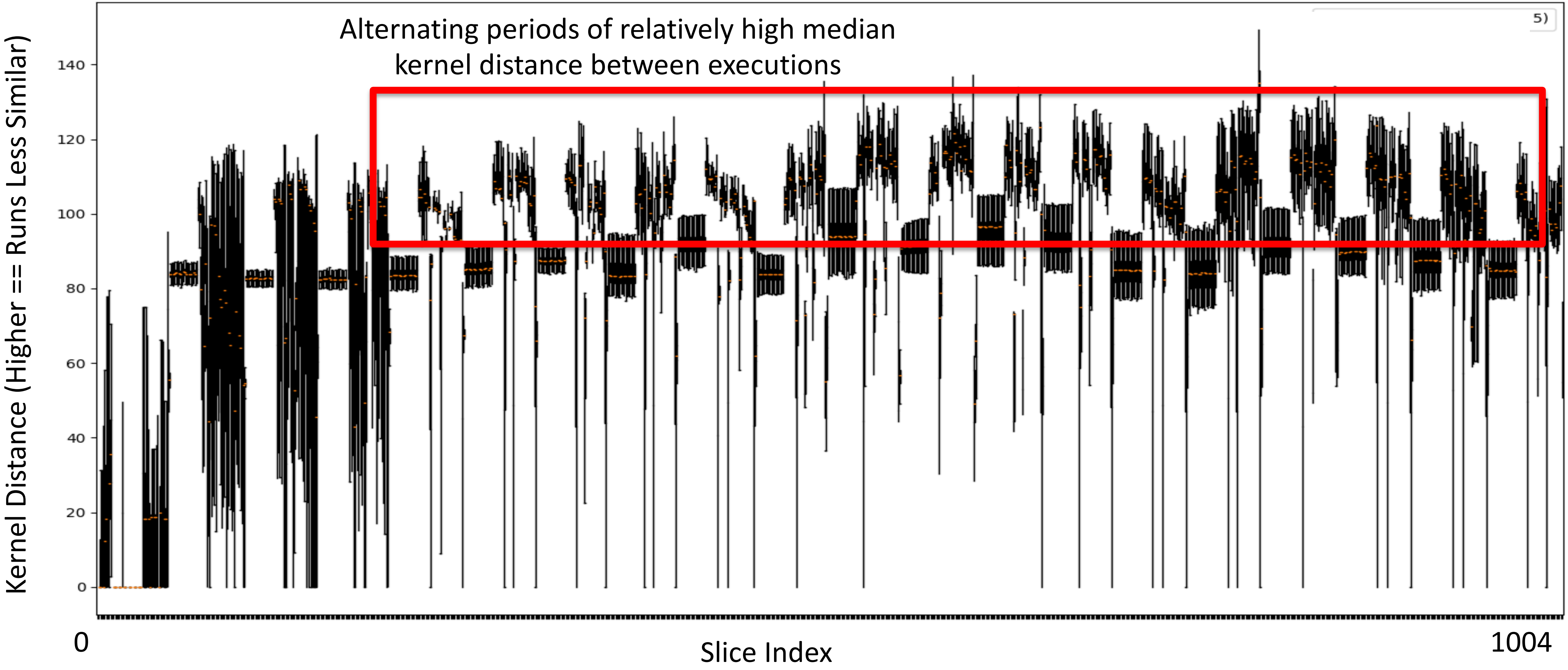
Kernel Distance Time Series for miniAMR



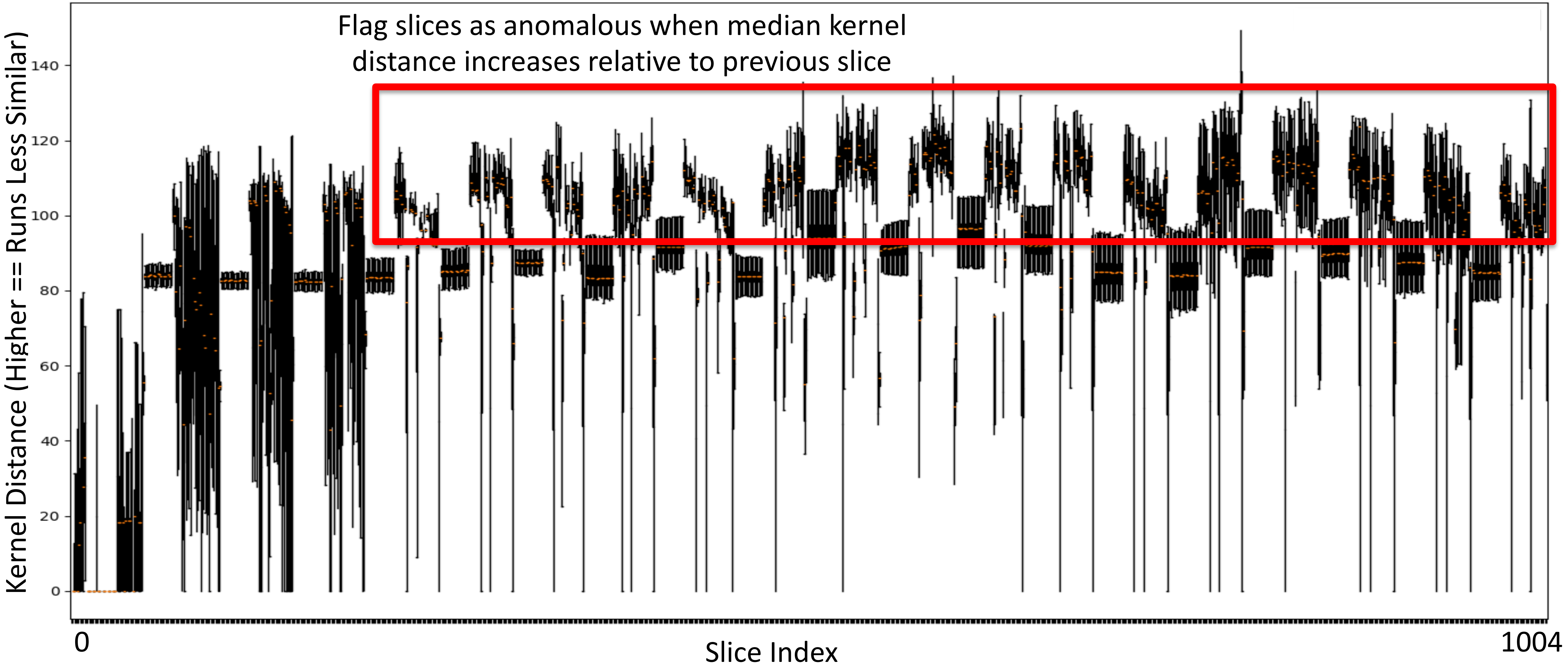
Kernel Distance Time Series for miniAMR



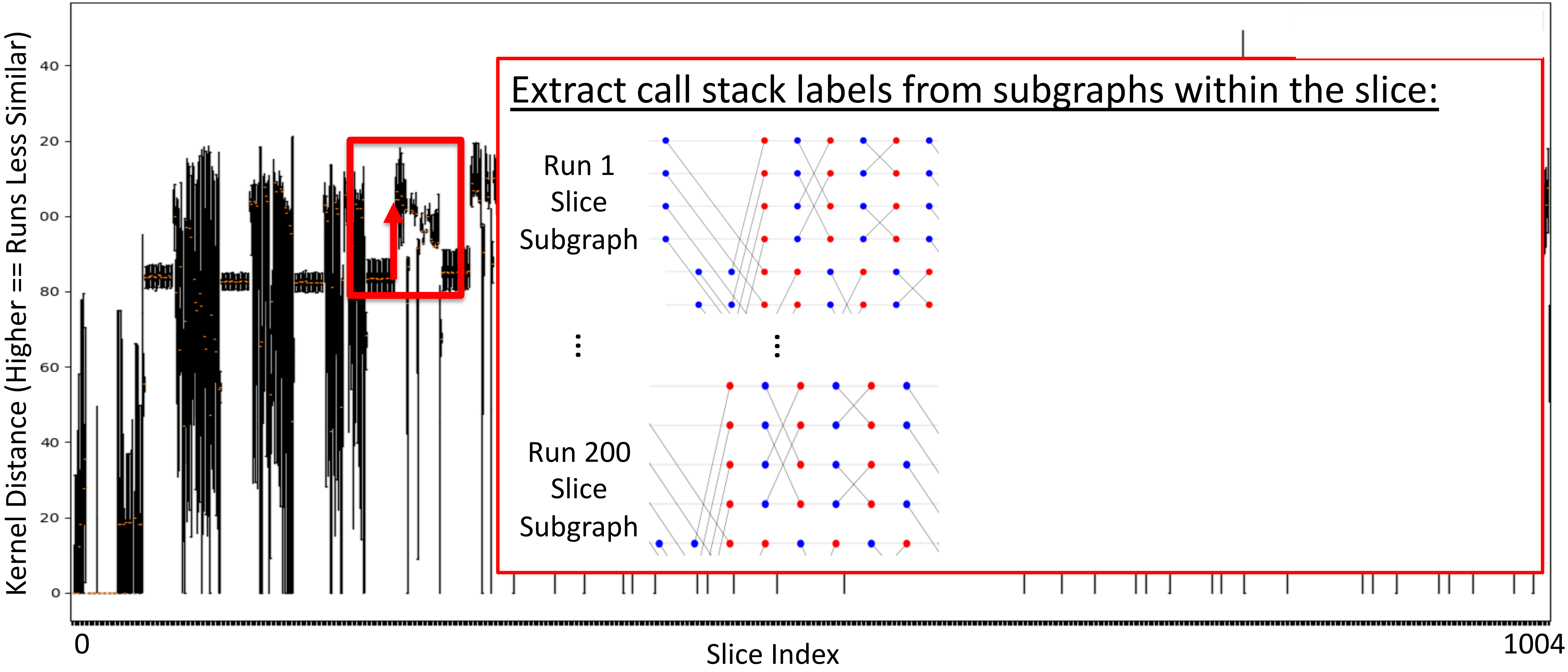
Identifying Anomalous Slices



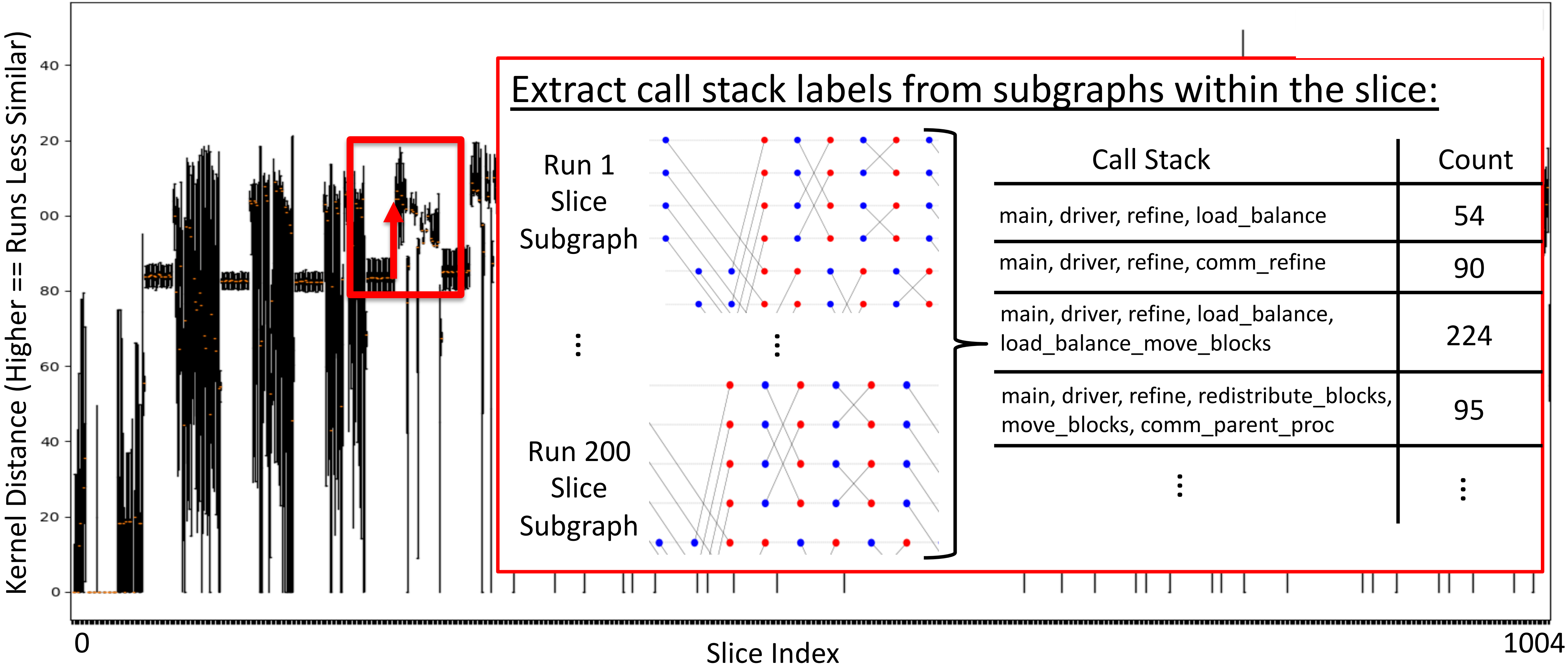
Identifying Anomalous Slices



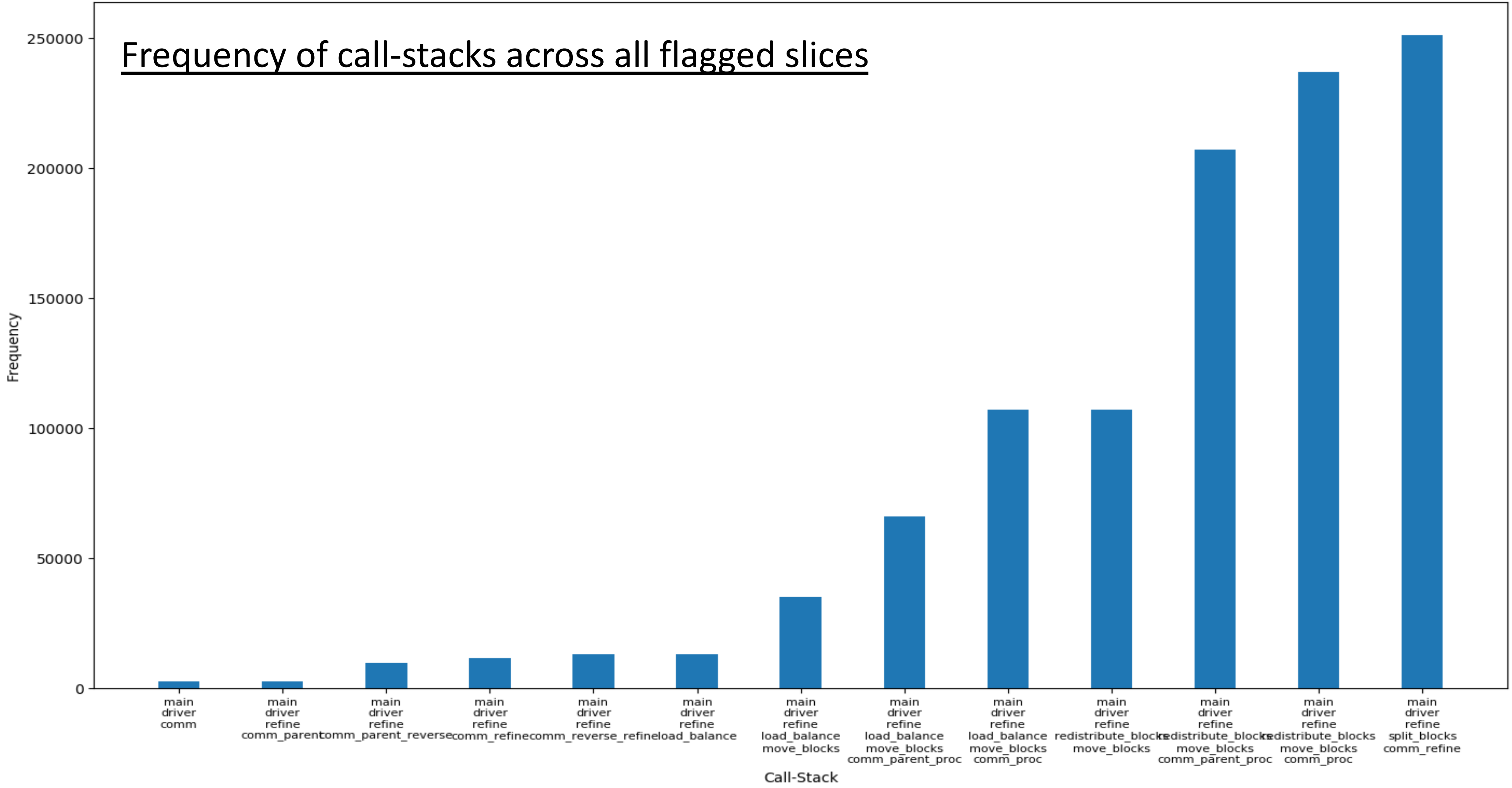
Linking to Potential Root Causes



Linking to Potential Root Causes



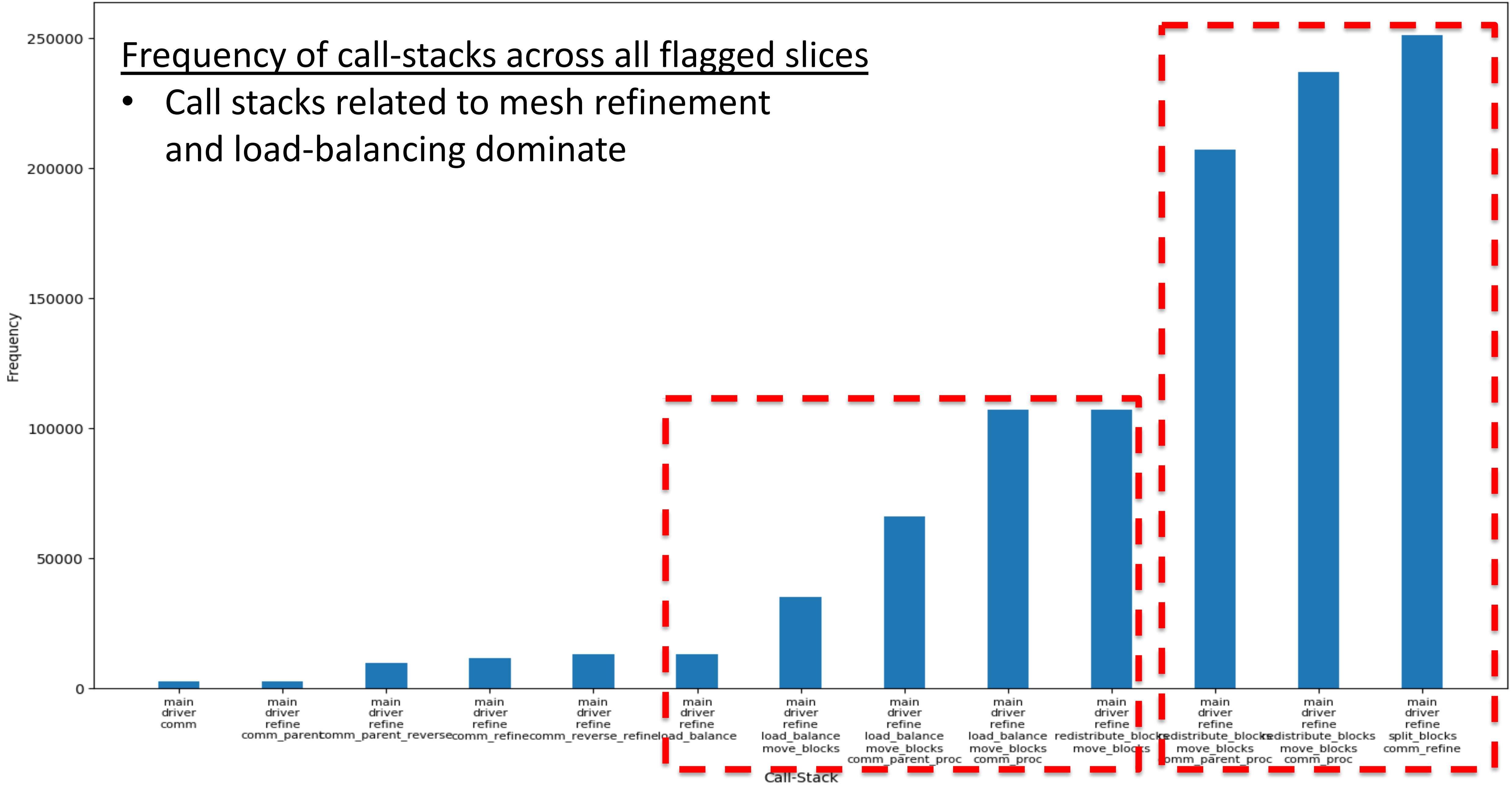
Call Stack	Count
main, driver, refine, load_balance	54
main, driver, refine, comm_refine	90
main, driver, refine, load_balance, load_balance_move_blocks	224
main, driver, refine, redistribute_blocks, move_blocks, comm_parent_proc	95
⋮	⋮



Frequency of call-stacks across all flagged slices
Frequency
250000
200000
150000
100000
50000
0
main driver comm
main driver refine comm_parent
main driver refine comm_parent_reverse
main driver refine comm_refine
main driver refine comm_reverse_refine
main driver refine load_balance
main driver refine load_balance move_blocks
main driver refine load_balance move_blocks comm_parent_proc
main driver refine load_balance move_blocks comm_proc
main driver refine redistribute_blocks move_blocks
main driver refine redistribute_blocks move_blocks comm_parent_proc
main driver refine redistribute_blocks move_blocks comm_proc
main driver refine split_blocks comm_refine
Call-Stack

Frequency of call-stacks across all flagged slices

- Call stacks related to mesh refinement and load-balancing dominate



Conclusion

- Our workflow identifies potential root causes of non-determinism in HPC applications by:
  - Building event graph models of executions
  - Quantifying trends in non-determinism via graph kernel distance
  - Linking runtime non-determinism to root causes

Conclusion

- Our workflow identifies potential root causes of non-determinism in HPC applications by:
  - Building event graph models of executions
  - Quantifying trends in non-determinism via graph kernel distance
  - Linking runtime non-determinism to root causes
- Demonstrated viability against:
  - Isolated communication patterns (CORAL-2 AMG)
  - Representative mini-app (miniAMR)

Conclusion

- Our workflow identifies potential root causes of non-determinism in HPC applications by:
  - Building event graph models of executions
  - Quantifying trends in non-determinism via graph kernel distance
  - Linking runtime non-determinism to root causes
- Demonstrated viability against:
  - Isolated communication patterns (CORAL-2 AMG)
  - Representative mini-app (miniAMR)
- Future Work:
  - Target full-fledged production AMR applications (e.g., Enzo)

Questions?

THE UNIVERSITY OF TENNESSEE KNOXVILLE

BIG ORANGE. BIG IDEAS.®