Scalable Offline Monitoring

Matúš Harvan

ABB Corporate Research

Joint work with David Basin, Germano Caronni, Sarah Ereth, Felix Klaedtke, and Heiko Mantel.

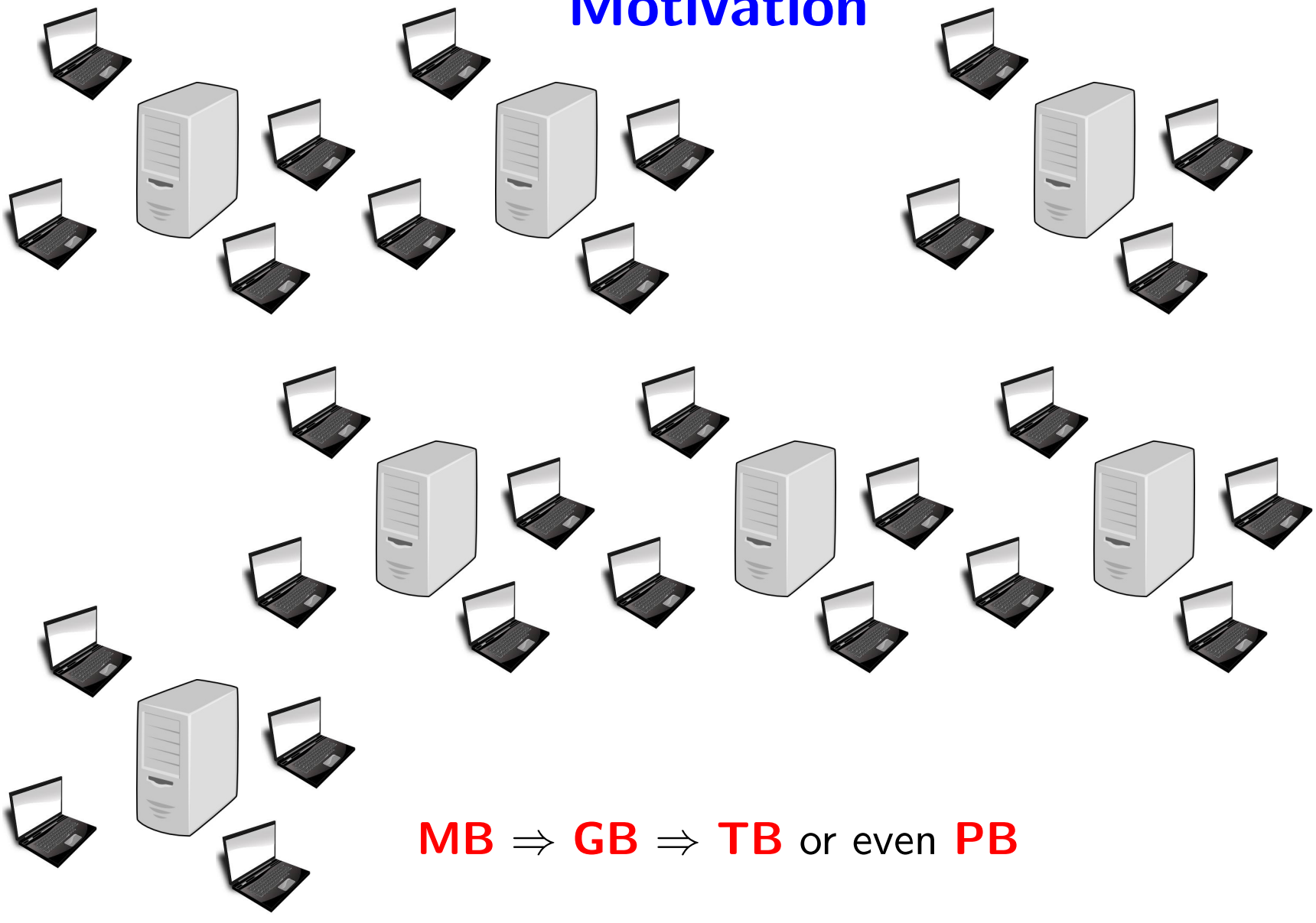
Motivation

MB

Motivation

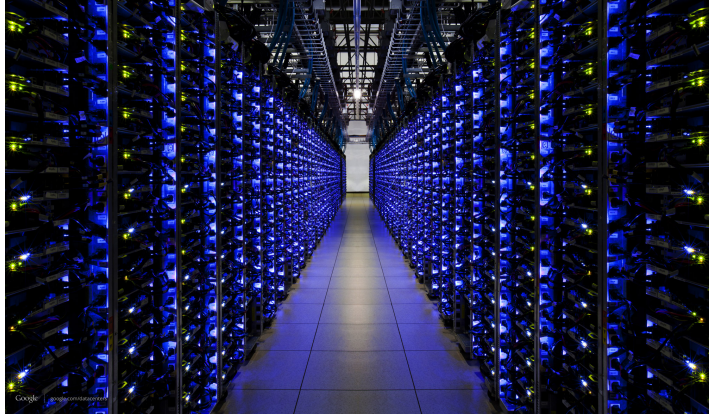
MB $\Rightarrow$ GB

Motivation



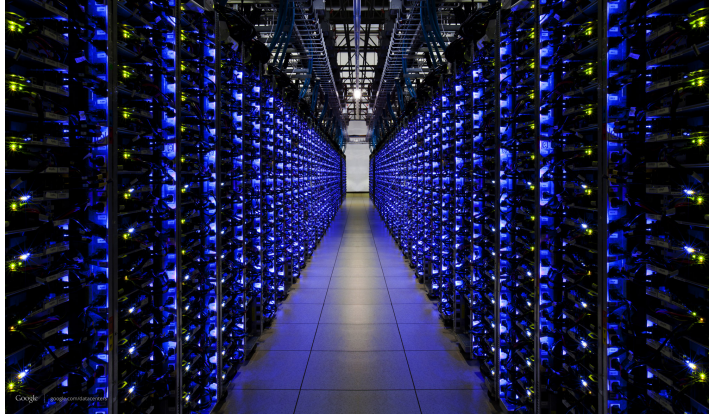
MB $\Rightarrow$ **GB** $\Rightarrow$ **TB** or even **PB**

Contributions



A solution to monitor big data

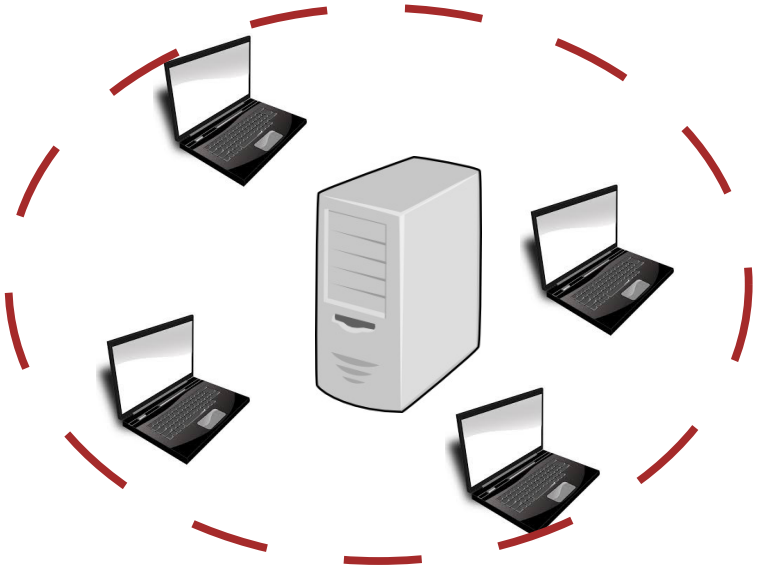
Contributions



A solution to monitor big data

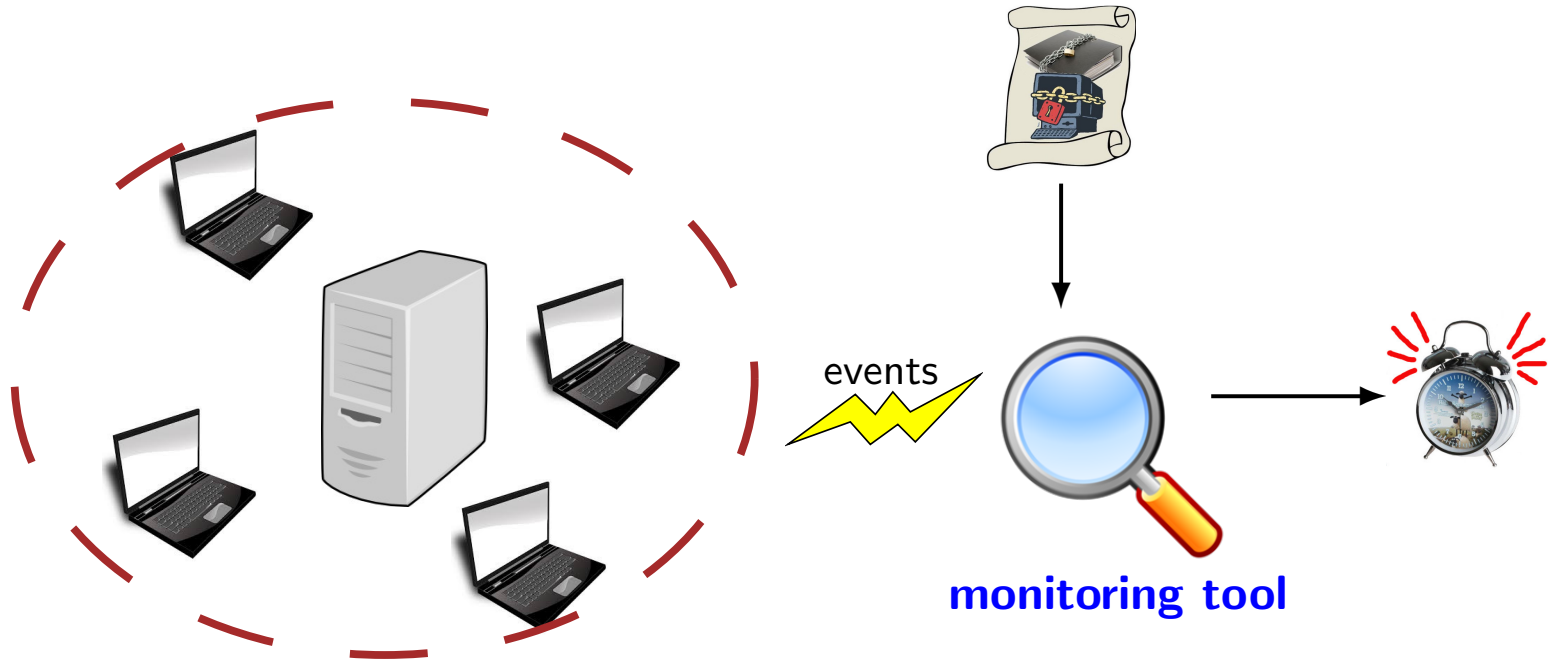
- Framework for parallelizing monitoring
  - Operators to slice logs.
  - Soundness & completeness guarantees.
- Algorithmic realization with MapReduce.
- Evaluation on real-world data.

Setting and Prior Work



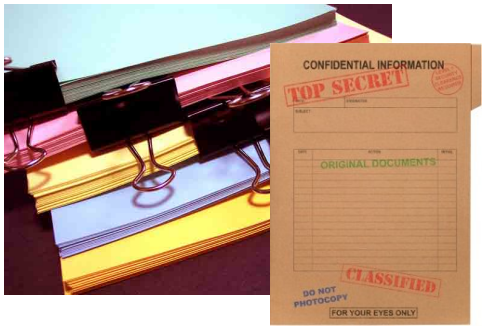
policies

Setting and Prior Work



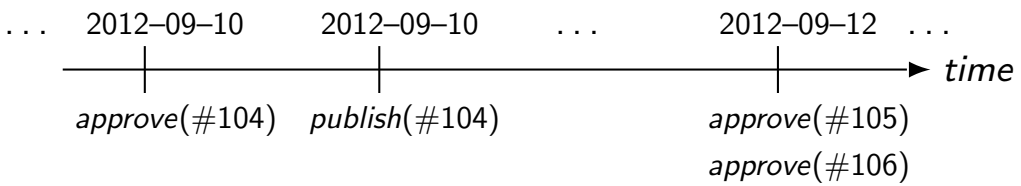
- General solution using metric first-order temporal logic (MFOTL).
- Monitoring algorithm implemented in Monpoly tool.

Example



- Policy:
  1. *Reports must be approved before they are published.*
  2. *Approvals must happen at most 10 days before publication.*
- Events are logged with time stamps:

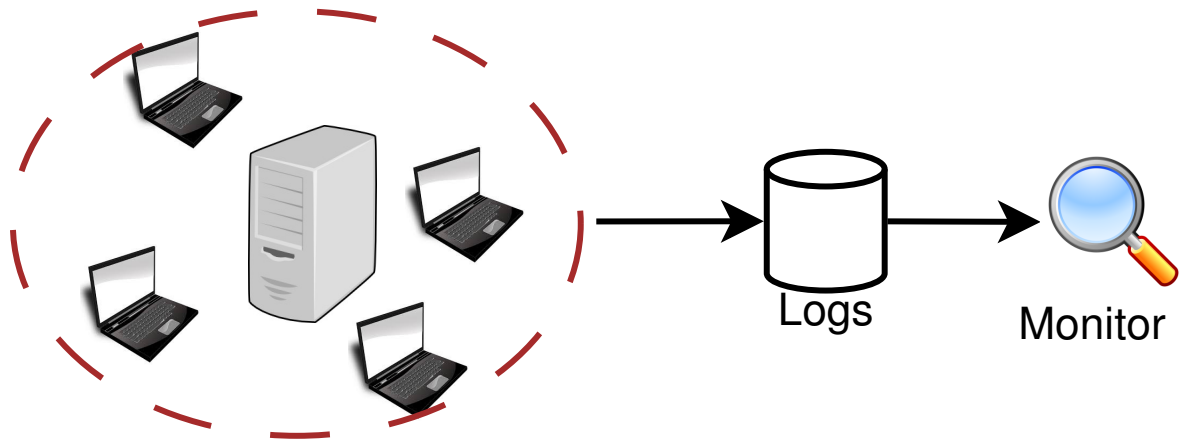
```
  ⋮              ⋮
2012-09-10    approve (#104)
2012-09-10    publish (#104)
  ⋮              ⋮
2012-09-12    approve (#105)
              approve (#106)
  ⋮              ⋮
```



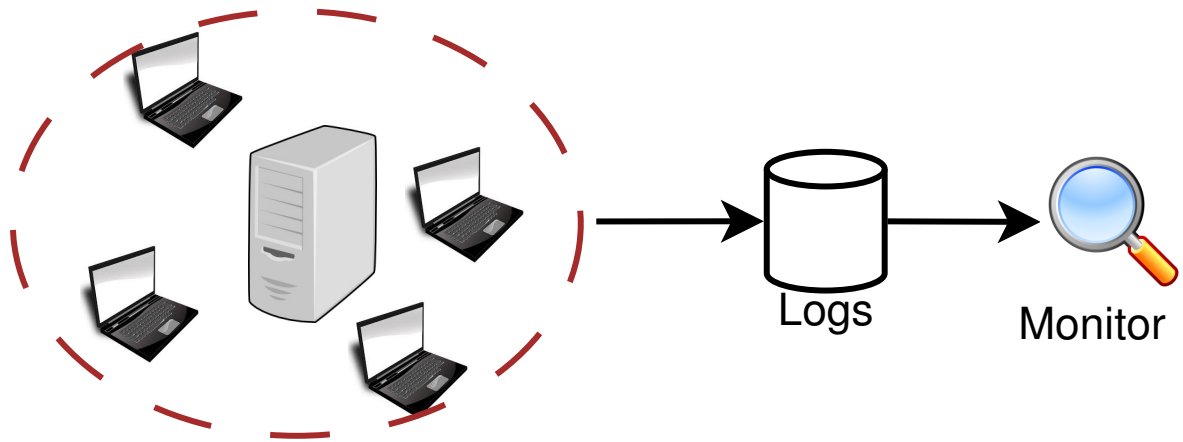
- Policy formalized in metric first-order temporal logic:

$$\Box\, \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$$

Framework to parallelize monitoring

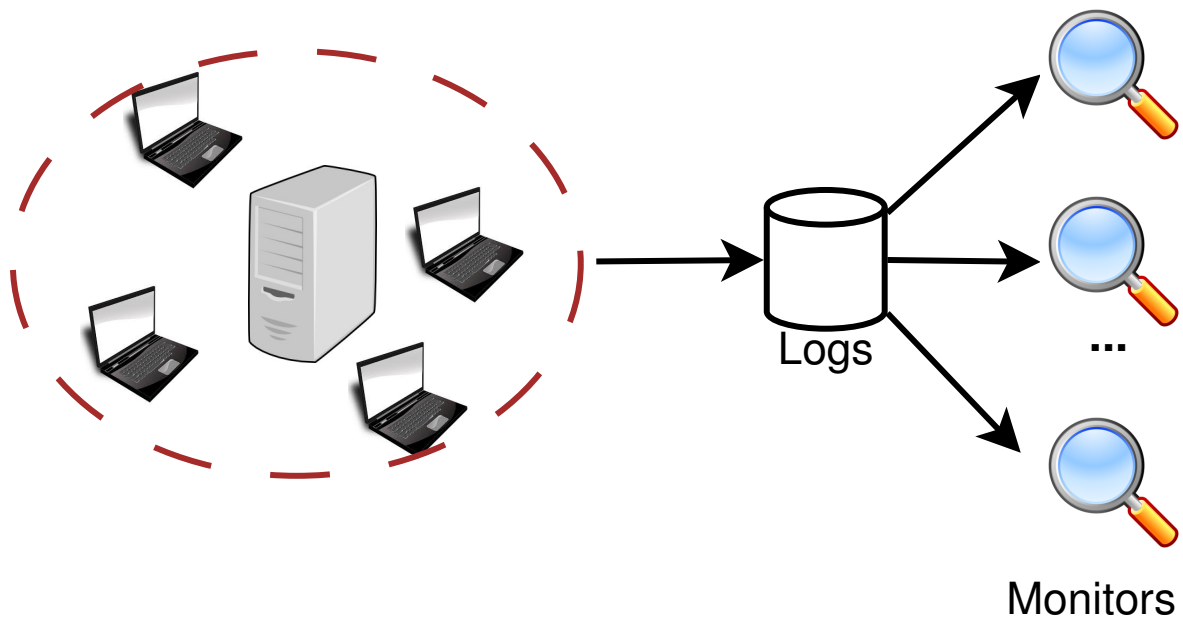


Framework to parallelize monitoring



Framework to parallelize monitoring

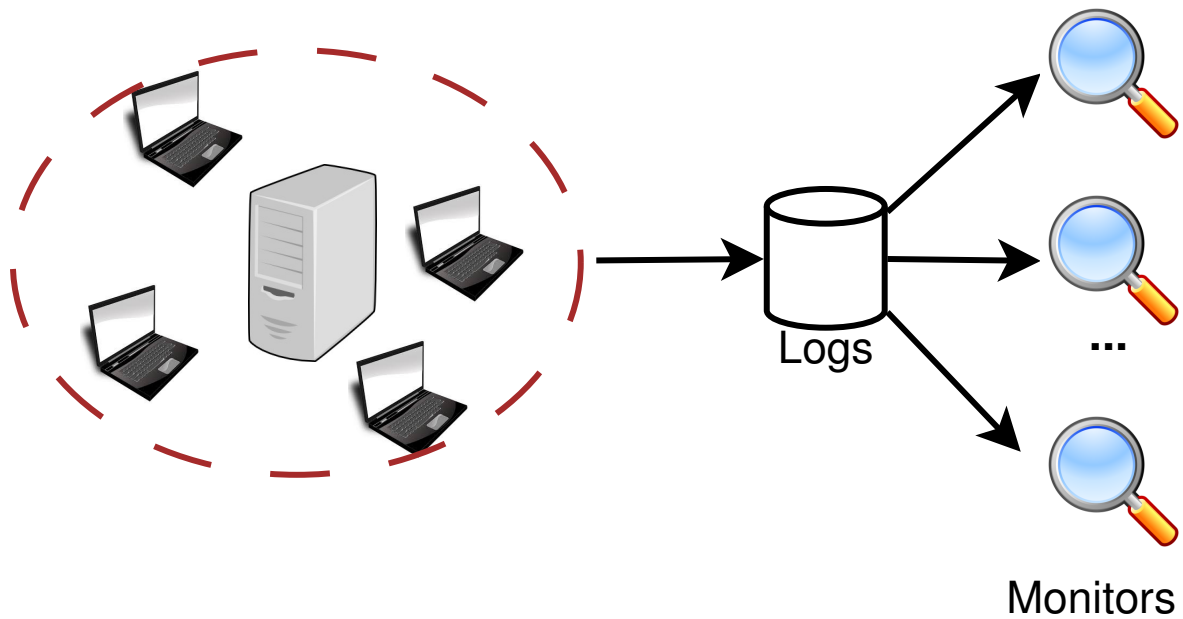
- Split a large log into smaller log parts (slices) that can be monitored independently of each other.



Framework to parallelize monitoring

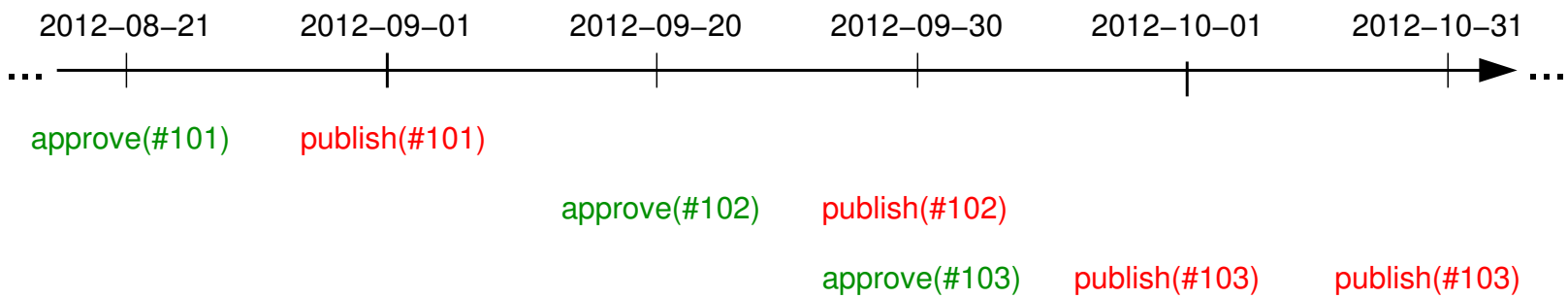
- Split a large log into smaller log parts (slices) that can be monitored independently of each other.

- Slicing methods:
  1. By time
  2. By data
  3. Filtering
- Compositionality
- Sound & complete



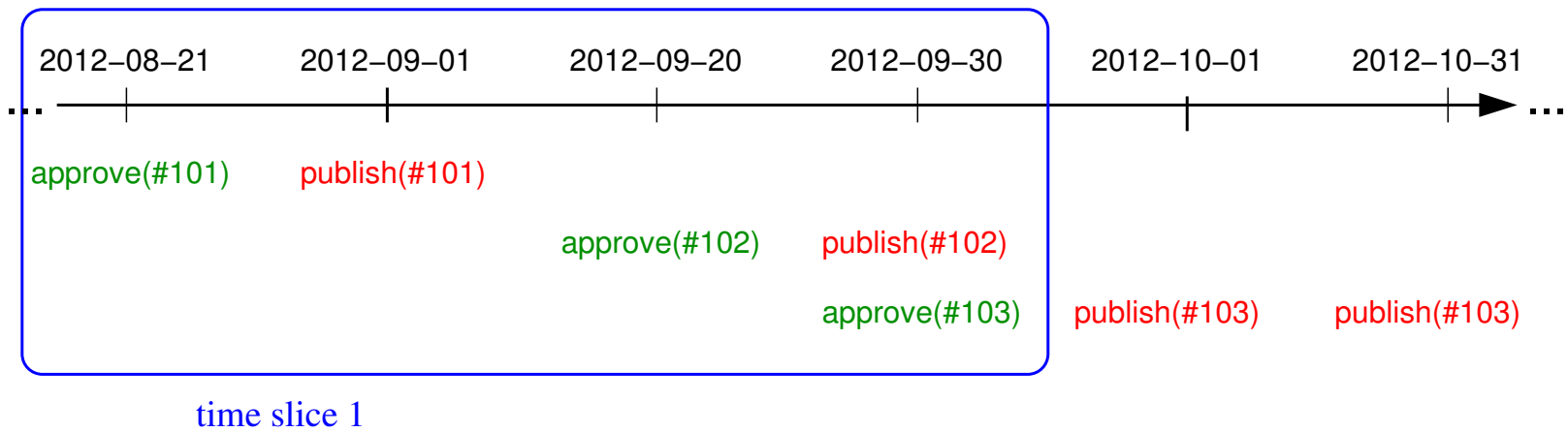
Slicing Time

- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



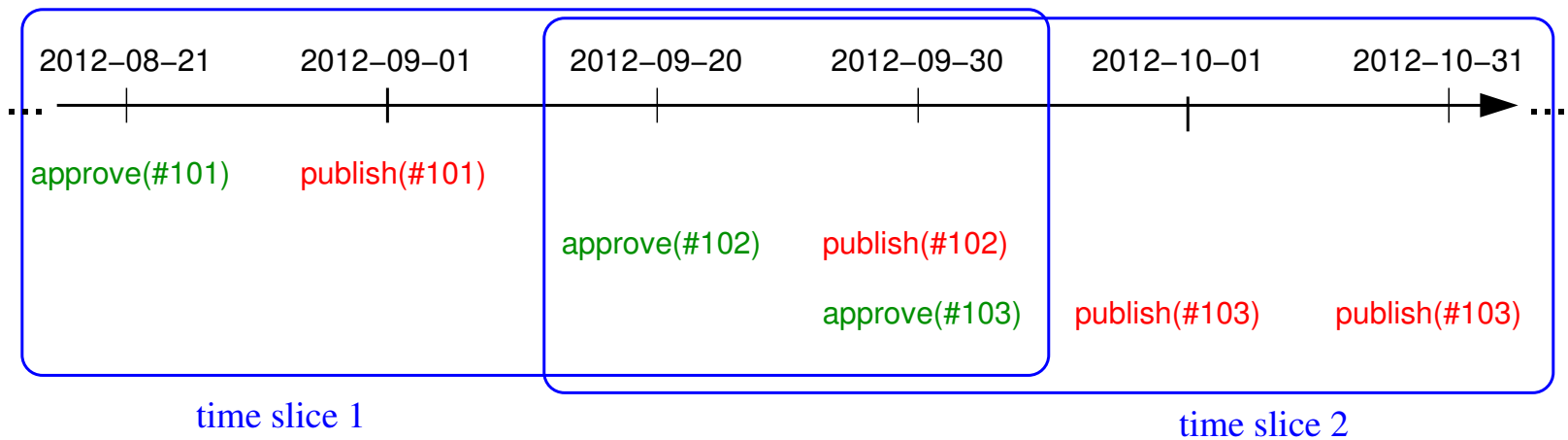
Slicing Time

- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



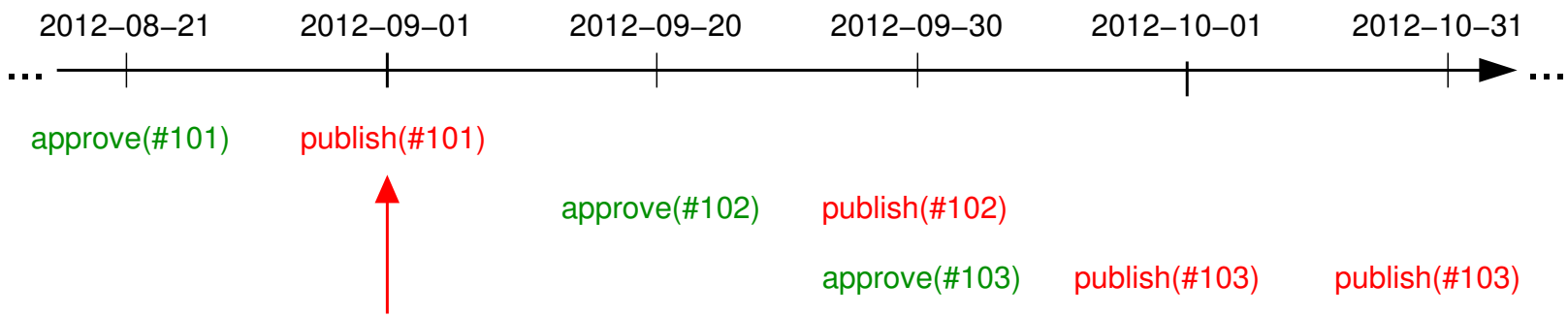
Slicing Time

- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



Slicing Time

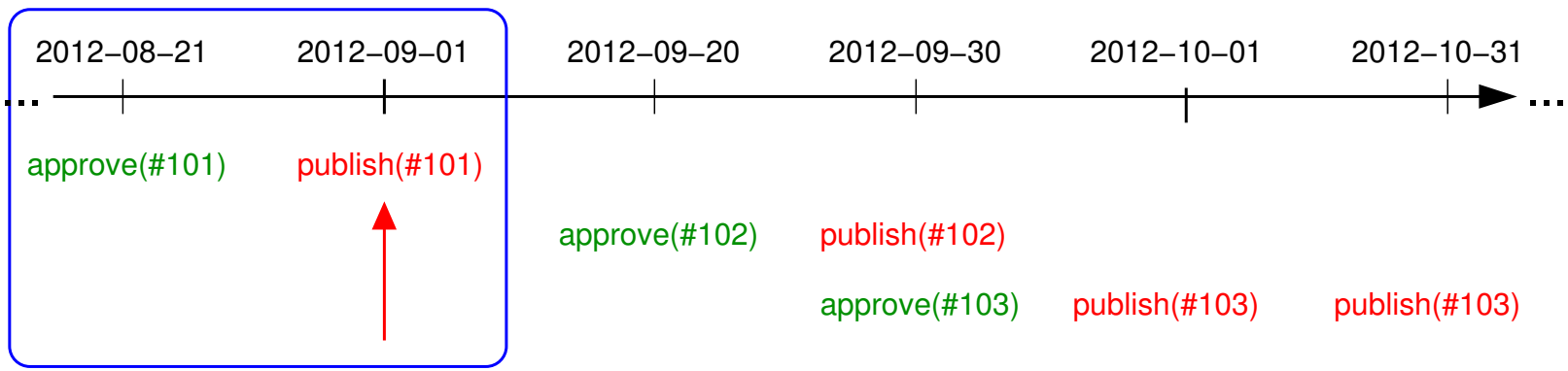
- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



- Relative interval determines required additional points.
  $\rightarrow$ Derived from temporal operators.
- Restrictions determine which violations are valid.

Slicing Time

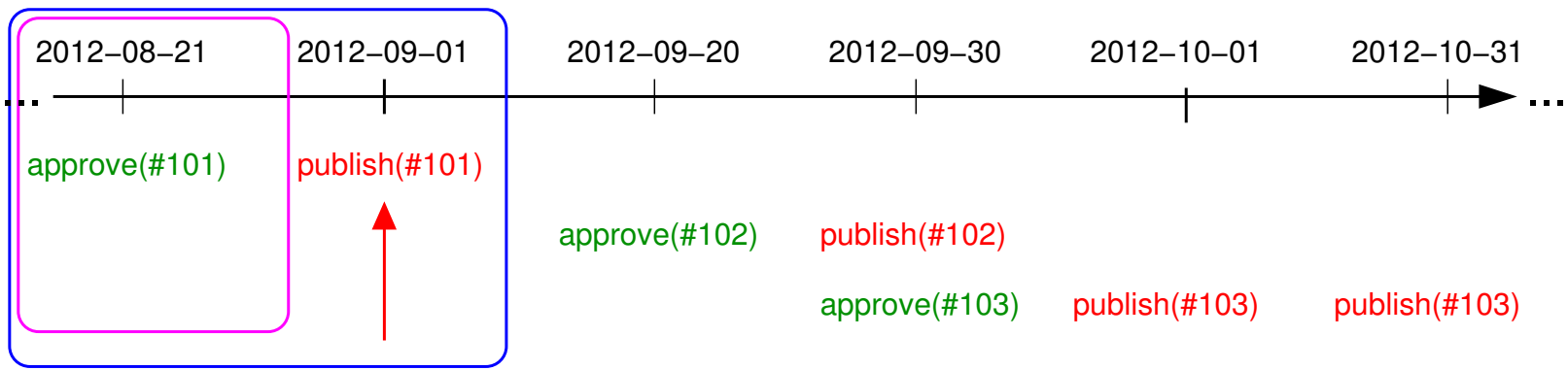
- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



- Relative interval determines required additional points.
  $\rightarrow$ Derived from temporal operators.
- Restrictions determine which violations are valid.

Slicing Time

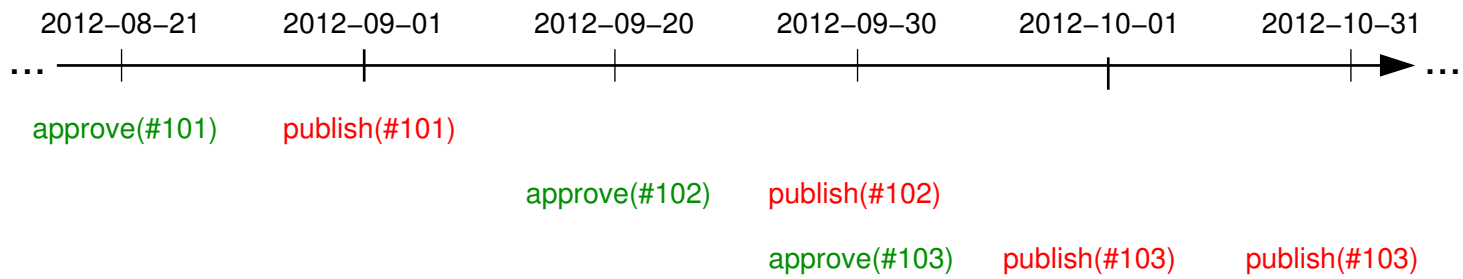
- Split log based on the timestamps.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different months.



- Relative interval determines required additional points.
  $\rightarrow$ Derived from temporal operators.
- Restrictions determine which violations are valid.

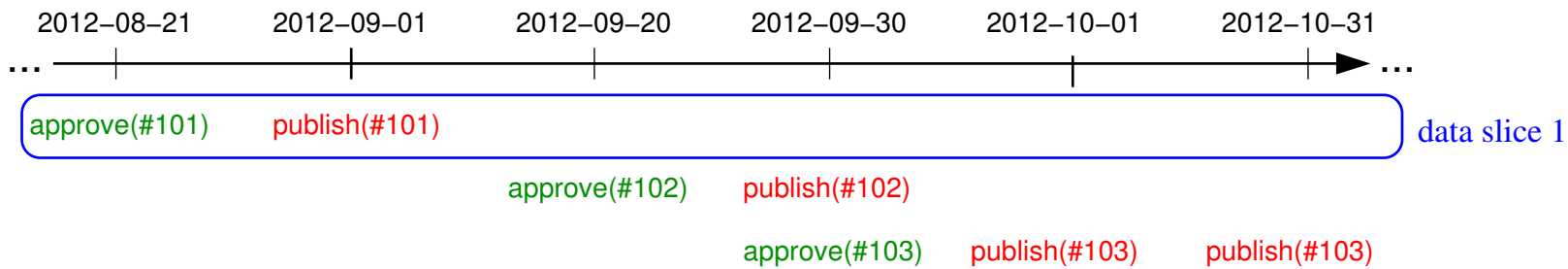
Slicing Data

- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



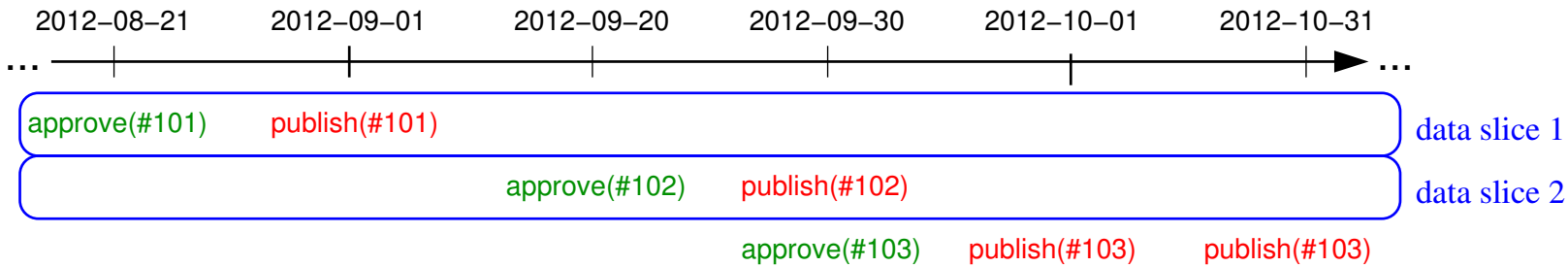
Slicing Data

- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



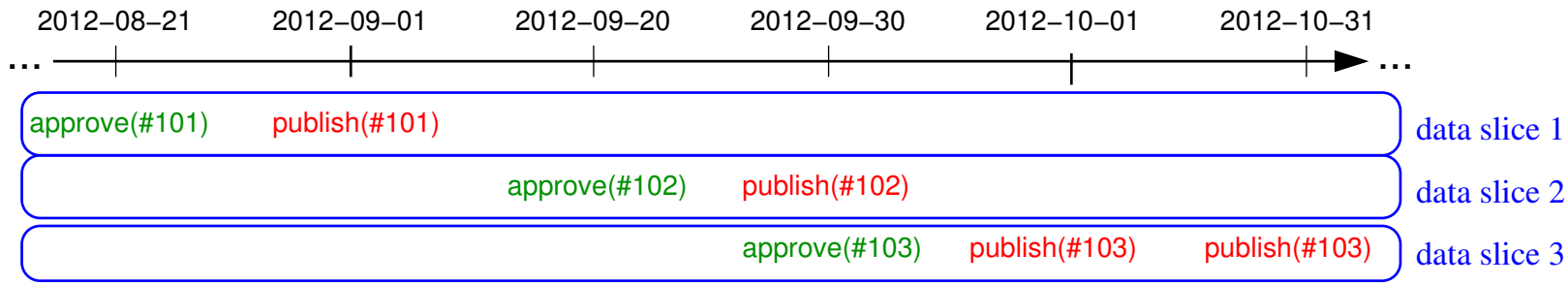
Slicing Data

- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



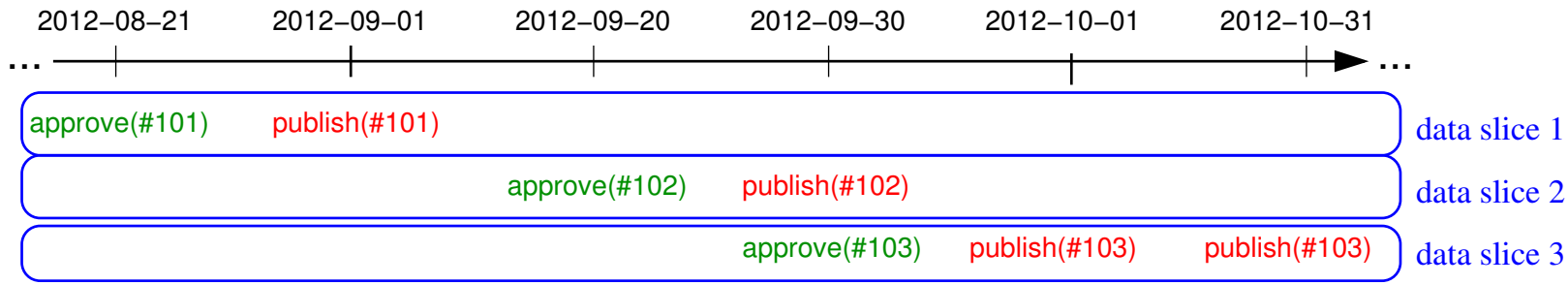
Slicing Data

- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



Slicing Data

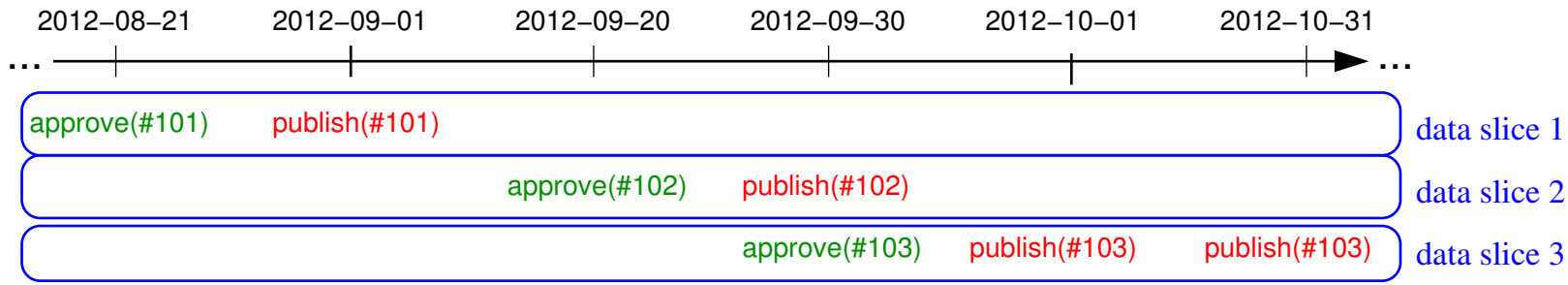
- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



- What about these formulas?
  - $\forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge\, \neg \exists r'.\, \mathit{publish}(r') \wedge r' > r$
  - $\forall r.\, \mathit{publish}(r) \rightarrow \Diamond_{\leq 7}\, \mathit{publish}(\mathrm{summary})$

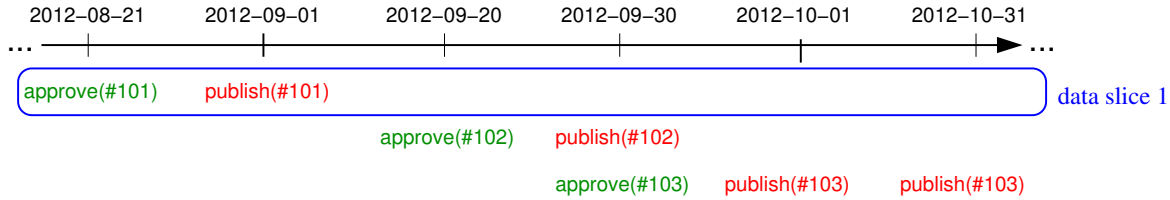
Slicing Data

- Split log based on parameters of log events.
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$
  - Slices cover different reports.



- What about these formulas?
  - $\forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge\, \neg \exists r'.\, \mathit{publish}(r') \wedge r' > r$
  - $\forall r.\, \mathit{publish}(r) \rightarrow \Diamond_{\leq 7}\, \mathit{publish}(\mathrm{summary})$
- How to do data slicing in general?
  - Trade-off: efficiency versus size.
  - Determine slice membership by inspecting log events independently.

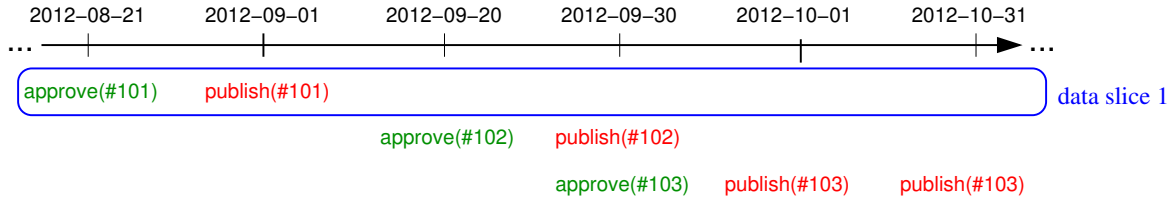
Slicing Data



- Choose slicing variable.
- Split domain into slicing sets.
  $\rightarrow$ Each slicing set induces a slice.
- Criteria whether log event $e(p_1, \ldots, p_n)$ belongs into slice:

  For each parameter p_i, look for atomic subformula $e(x_1, \ldots, x_n)$ where x_i is...

Slicing Data

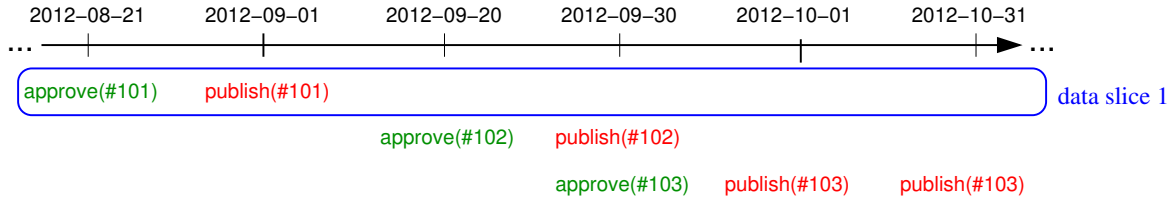


- Choose slicing variable.
- Split domain into slicing sets.
  $\rightarrow$ Each slicing set induces a slice.
- Criteria whether log event $e(p_1, \ldots, p_n)$ belongs into slice:

  For each parameter p_i, look for atomic subformula $e(x_1, \ldots, x_n)$ where x_i is...

  1. the slicing variable and p_i is in slicing set,
     Example: $\forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 10}\, approve(r)$

Slicing Data



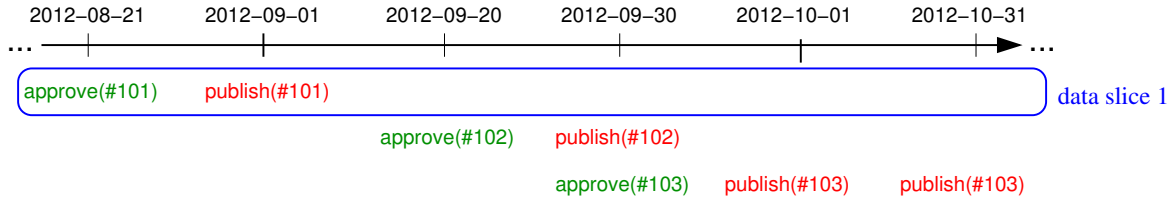
- Choose slicing variable.
- Split domain into slicing sets.
  $\rightarrow$ Each slicing set induces a slice.
- Criteria whether log event $e(p_1, \ldots, p_n)$ belongs into slice:

  For each parameter p_i, look for atomic subformula $e(x_1, \ldots, x_n)$ where x_i is...

  1. the slicing variable and p_i is in slicing set,
     Example: $\forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 10}\, approve(r)$

  2. another variable,
     Example: $\forall r.\, publish(r) \rightarrow \blacklozenge\, \neg\exists r'.\, publish(r') \wedge r' > r$

Slicing Data



- Choose slicing variable.
- Split domain into slicing sets.
  $\rightarrow$ Each slicing set induces a slice.
- Criteria whether log event $e(p_1, \ldots, p_n)$ belongs into slice:

  For each parameter p_i, look for atomic subformula $e(x_1, \ldots, x_n)$ where x_i is...

  1. the slicing variable and p_i is in slicing set,
     Example: $\forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 10}\, approve(r)$
  2. another variable,
     Example: $\forall r.\, publish(r) \rightarrow \blacklozenge\, \neg\exists r'.\, publish(r') \wedge r' > r$
  3. or a constant.
     Example: $\forall r.\, publish(r) \rightarrow \Diamond_{\leq 7}\, publish(\text{summary})$

Choice of Slicing Variable Matters

- Policy: $\Box \forall c.\, \forall s.\, \mathit{login}(c, s) \rightarrow \Diamond_{\leq 6}\, \mathit{notify}(0, s)$
- Log: $\mathit{login}(1,1), \mathit{login}(2,2), \mathit{login}(3,3), \mathit{login}(4,4)$
  $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$
- Slicing by variable c
  - Slicing set: $\{1, 2\}$
    Slice: $\mathit{login}(1,1), \mathit{login}(2,2),$
    $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$
  - Slicing set: $\{3, 4\}$
    Slice: $\mathit{login}(3,3), \mathit{login}(4,4),$
    $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$

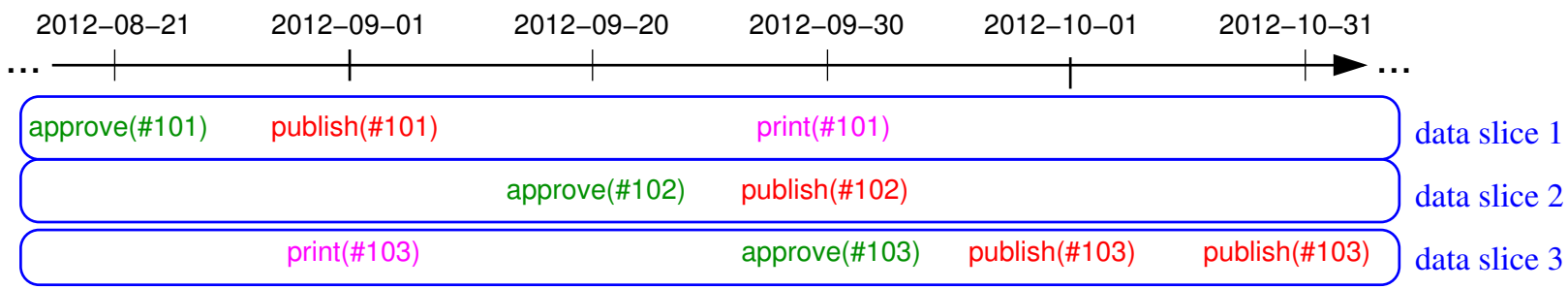
Choice of Slicing Variable Matters

- Policy: $\Box \forall c.\, \forall s.\, \mathit{login}(c, s) \rightarrow \Diamond_{\leq 6}\, \mathit{notify}(0, s)$
- Log: $\mathit{login}(1,1), \mathit{login}(2,2), \mathit{login}(3,3), \mathit{login}(4,4)$
  $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$
- Slicing by variable c
  - Slicing set: $\{1,2\}$
    Slice: $\mathit{login}(1,1), \mathit{login}(2,2),$
    $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$
  - Slicing set: $\{3,4\}$
    Slice: $\mathit{login}(3,3), \mathit{login}(4,4),$
    $\mathit{notify}(0,1), \mathit{notify}(0,2), \mathit{notify}(0,3), \mathit{notify}(0,4)$
- Slicing by variable s
  - Slicing set: $\{1,2\}$
    Slice: $\mathit{login}(1,1), \mathit{login}(1,2),$
    $\mathit{notify}(0,1), \mathit{notify}(0,2),$
  - Slicing set: $\{3,4\}$
    Slice: $\mathit{login}(3,3), \mathit{login}(4,4),$
    $\mathit{notify}(0,3), \mathit{notify}(0,4)$

Filtering

- Discards log parts "irrelevant" to the formula:
  1. "Irrelevant" *log events*.
     - Already discarded by data slicing.

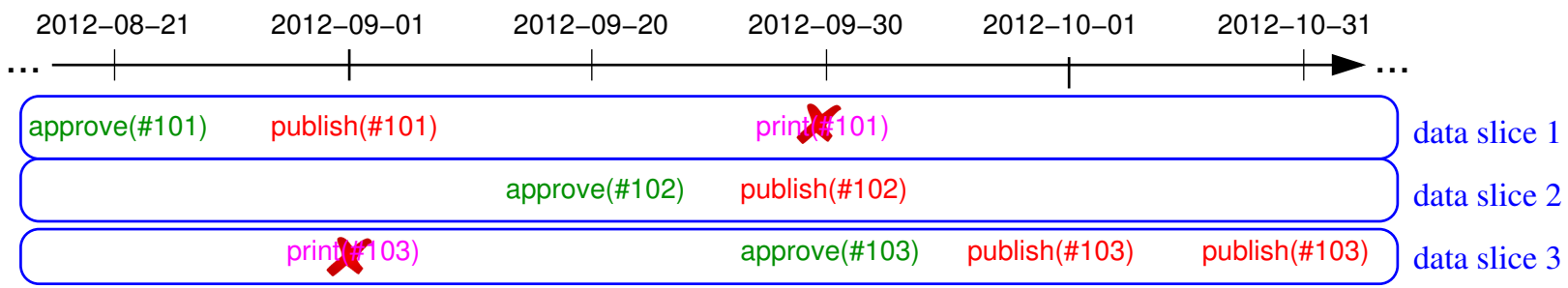
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10}\, \mathit{approve}(r)$



Filtering

- Discards log parts "irrelevant" to the formula:
  1. "Irrelevant" *log events*.
     - Already discarded by data slicing.

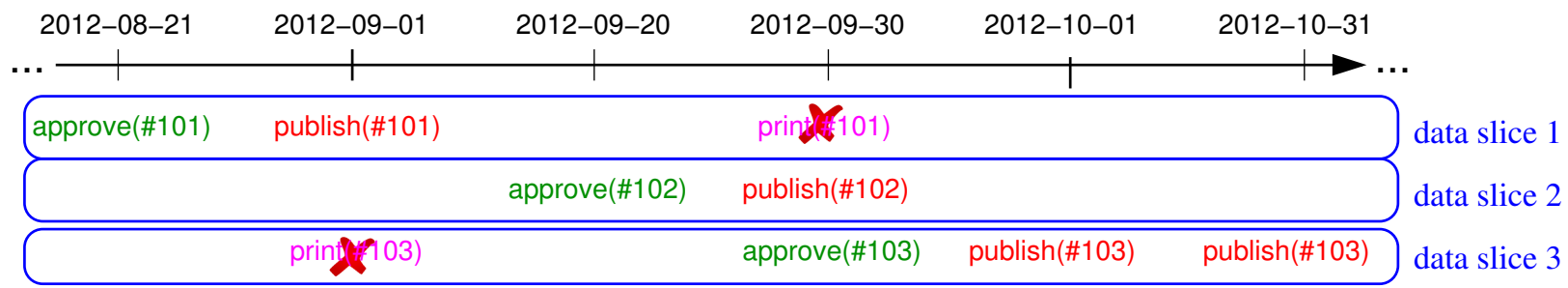
- Example: $\Box \forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 10}\, approve(r)$



Filtering

- Discards log parts "irrelevant" to the formula:
  1. "Irrelevant" *log events*.
     - Already discarded by data slicing.
  2. *Empty time points*.
     - Can safely be done only for some formulas.
       Problematic subformula examples:
       $\Box \forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 5} \blacklozenge_{\leq 5} approve(r)$
       $\Box \forall r.\, publish(r) \rightarrow \Box_{\leq 60} \neg product\_launch(r)$
     - Safe formulas approximated by a syntactically-defined fragment.

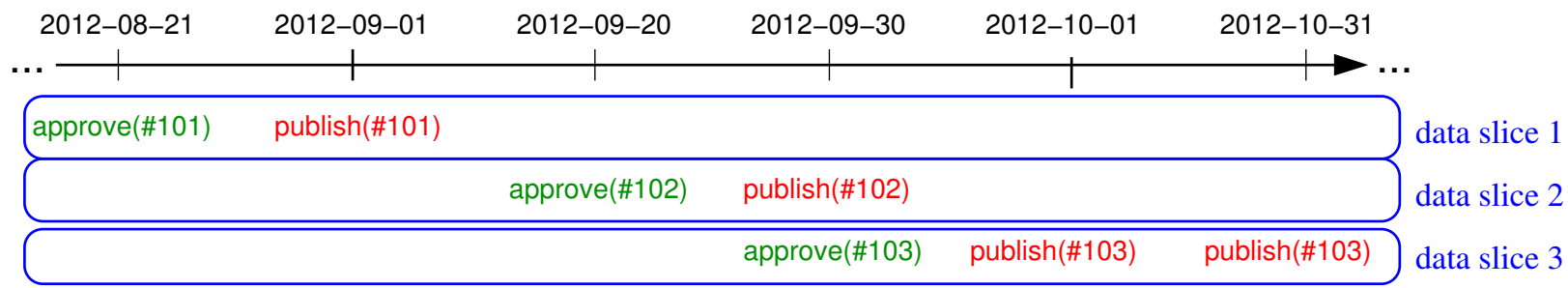
- Example: $\Box \forall r.\, publish(r) \rightarrow \blacklozenge_{\leq 10} approve(r)$



Filtering

- Discards log parts "irrelevant" to the formula:
  1. "Irrelevant" *log events*.
     - Already discarded by data slicing.
  2. *Empty time points*.
     - Can safely be done only for some formulas.
       Problematic subformula examples:
       $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 5} \blacklozenge_{\leq 5} \mathit{approve}(r)$
       $\Box \forall r.\, \mathit{publish}(r) \rightarrow \Box_{\leq 60} \neg \mathit{product\_launch}(r)$
     - Safe formulas approximated by a syntactically-defined fragment.

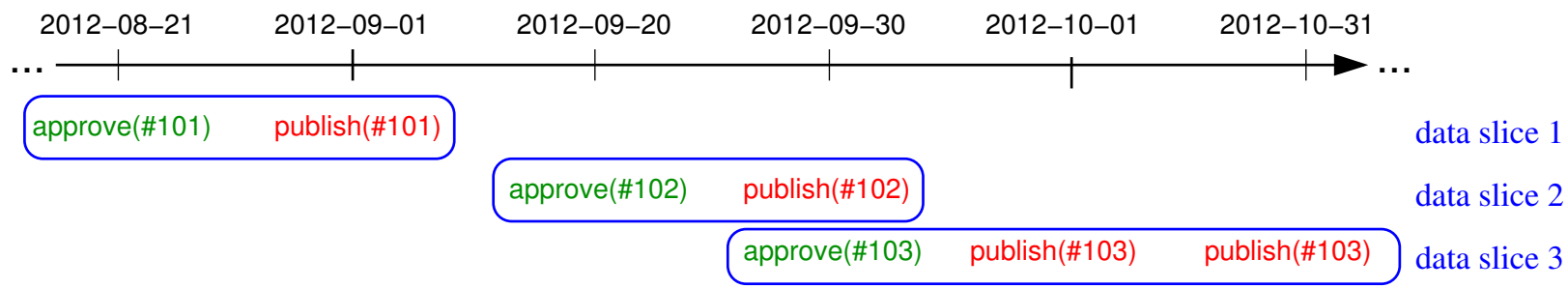
- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10} \mathit{approve}(r)$



Filtering

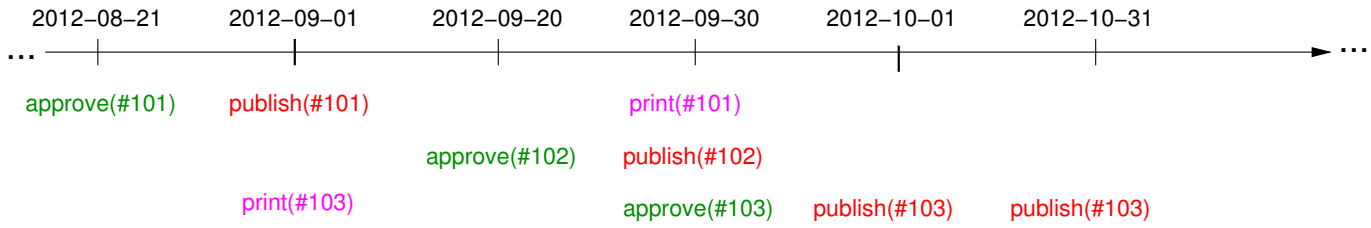
- Discards log parts "irrelevant" to the formula:
  1. "Irrelevant" *log events*.
     - Already discarded by data slicing.
  2. *Empty time points*.
     - Can safely be done only for some formulas.
       Problematic subformula examples:
       $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 5} \blacklozenge_{\leq 5} \mathit{approve}(r)$
       $\Box \forall r.\, \mathit{publish}(r) \rightarrow \Box_{\leq 60} \neg \mathit{product\_launch}(r)$
     - Safe formulas approximated by a syntactically-defined fragment.

- Example: $\Box \forall r.\, \mathit{publish}(r) \rightarrow \blacklozenge_{\leq 10} \mathit{approve}(r)$



Composition

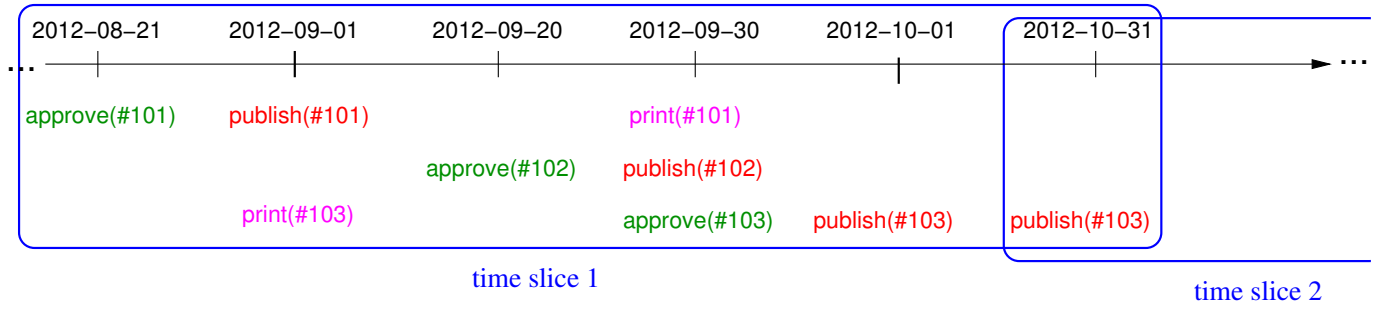
- Combine slicing methods arbitrarily.



- Sound & complete by construction.

Composition

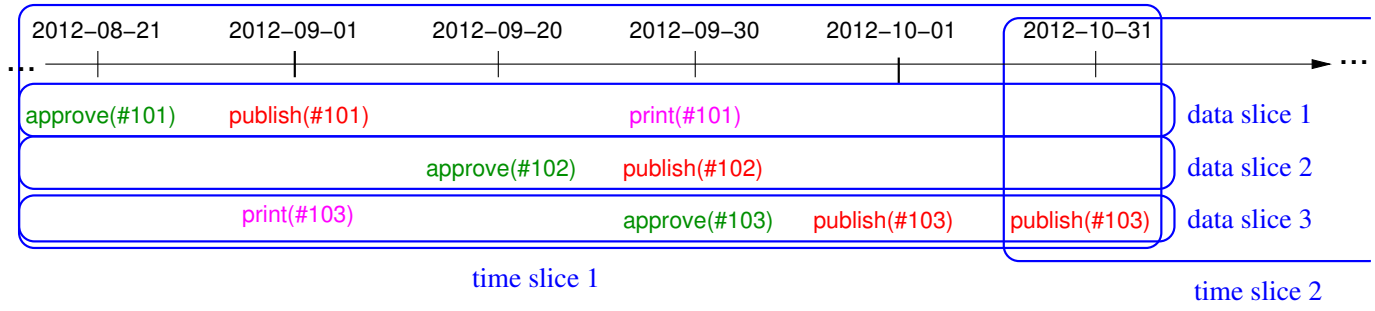
- Combine slicing methods arbitrarily.
  - Slice time



- Sound & complete by construction.

Composition

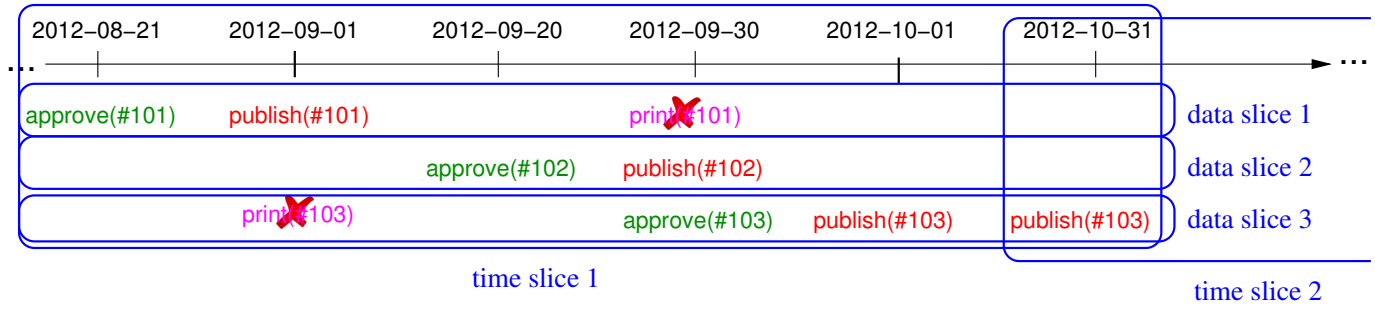
- Combine slicing methods arbitrarily.
  - Slice time
  - Slice data



- Sound & complete by construction.

Composition

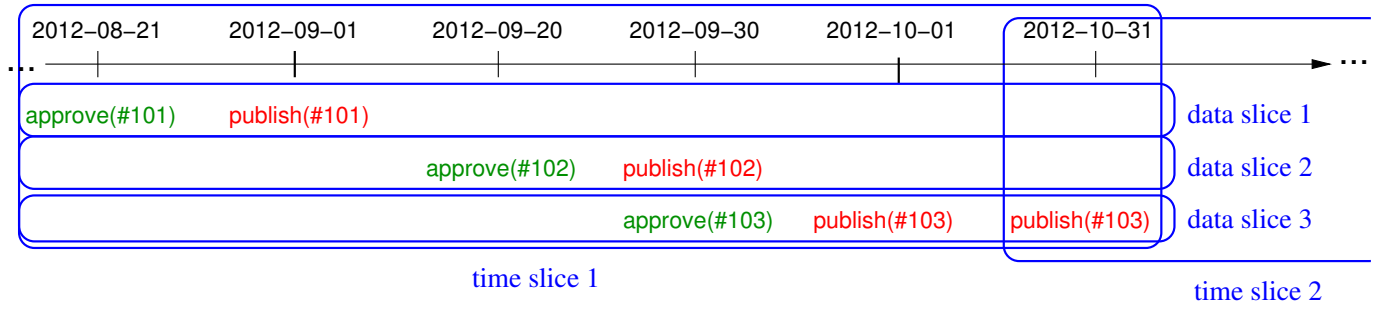
- Combine slicing methods arbitrarily.
  - Slice time
  - Slice data
  - Filter "irrelevant" log events.



- Sound & complete by construction.

Composition

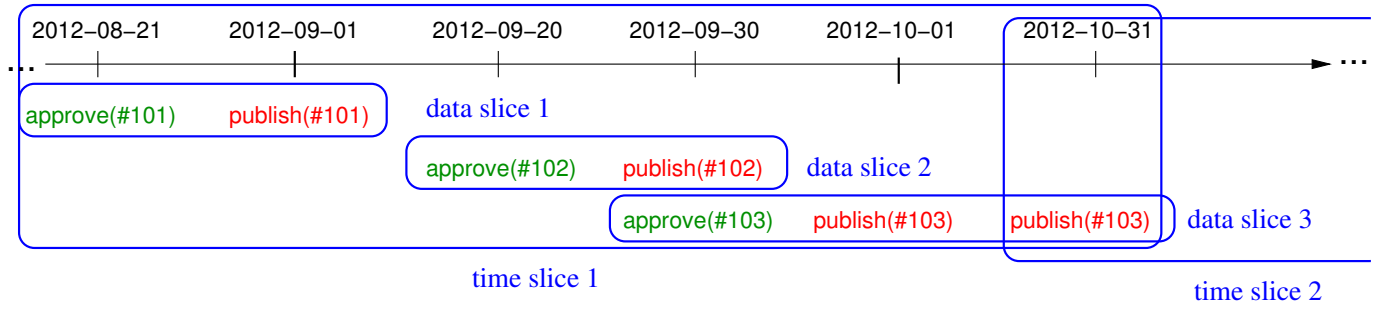
- Combine slicing methods arbitrarily.
  - Slice time
  - Slice data
  - Filter "irrelevant" log events.



- Sound & complete by construction.

Composition

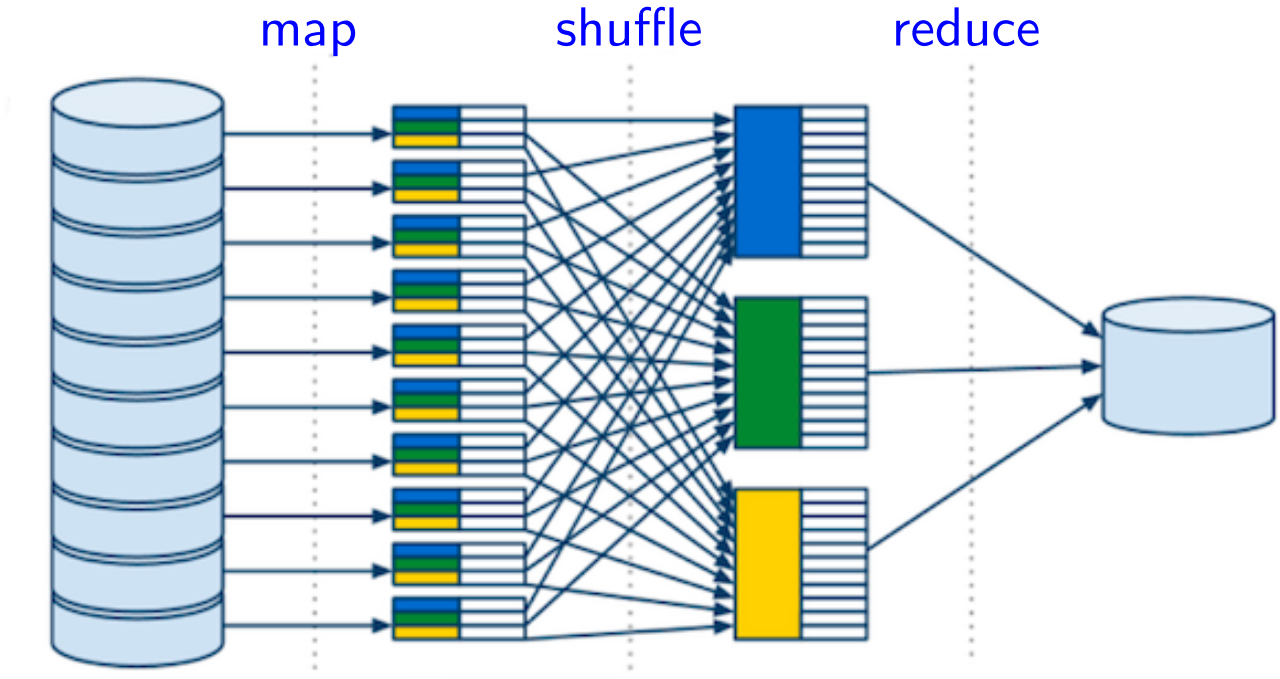
- Combine slicing methods arbitrarily.
  - Slice time
  - Slice data
  - Filter "irrelevant" log events.
  - Filter empty time points.



- Sound & complete by construction.

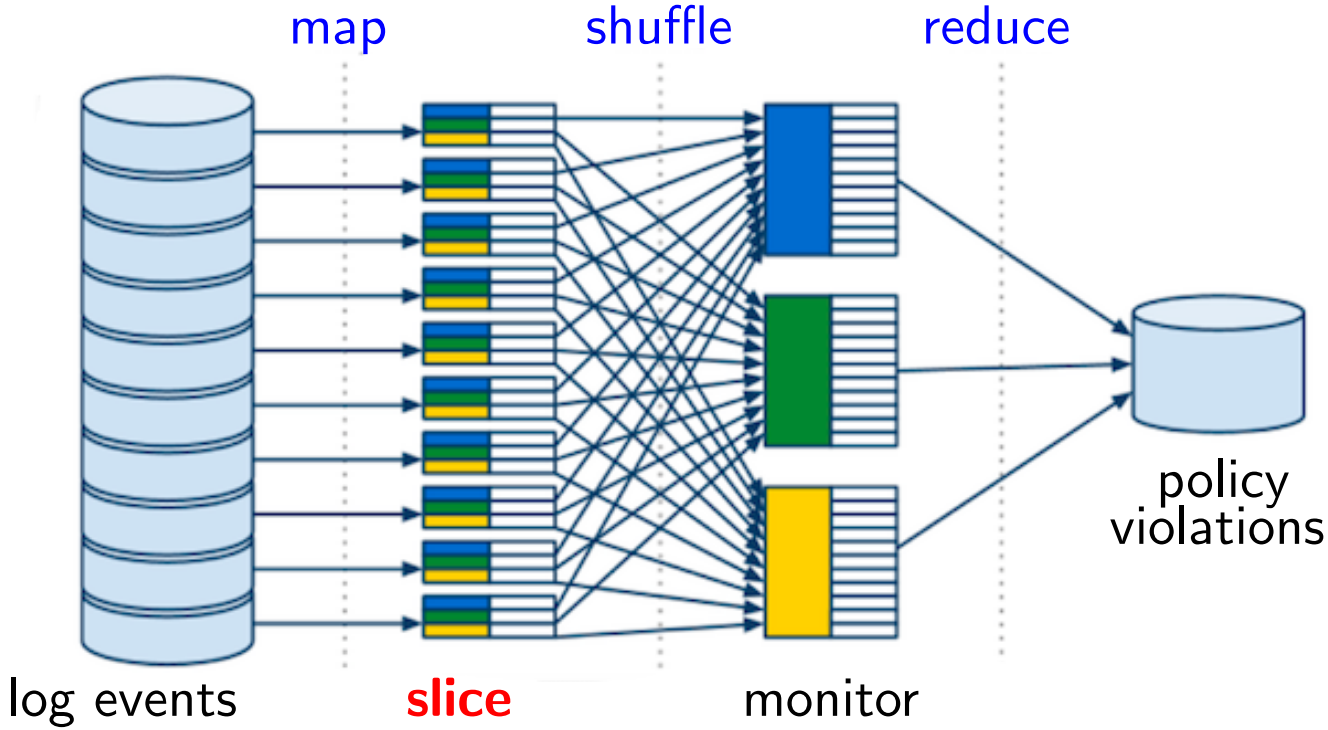
Algorithmic Realization with MapReduce

- MapReduce framework supports data-intensive distributed computations



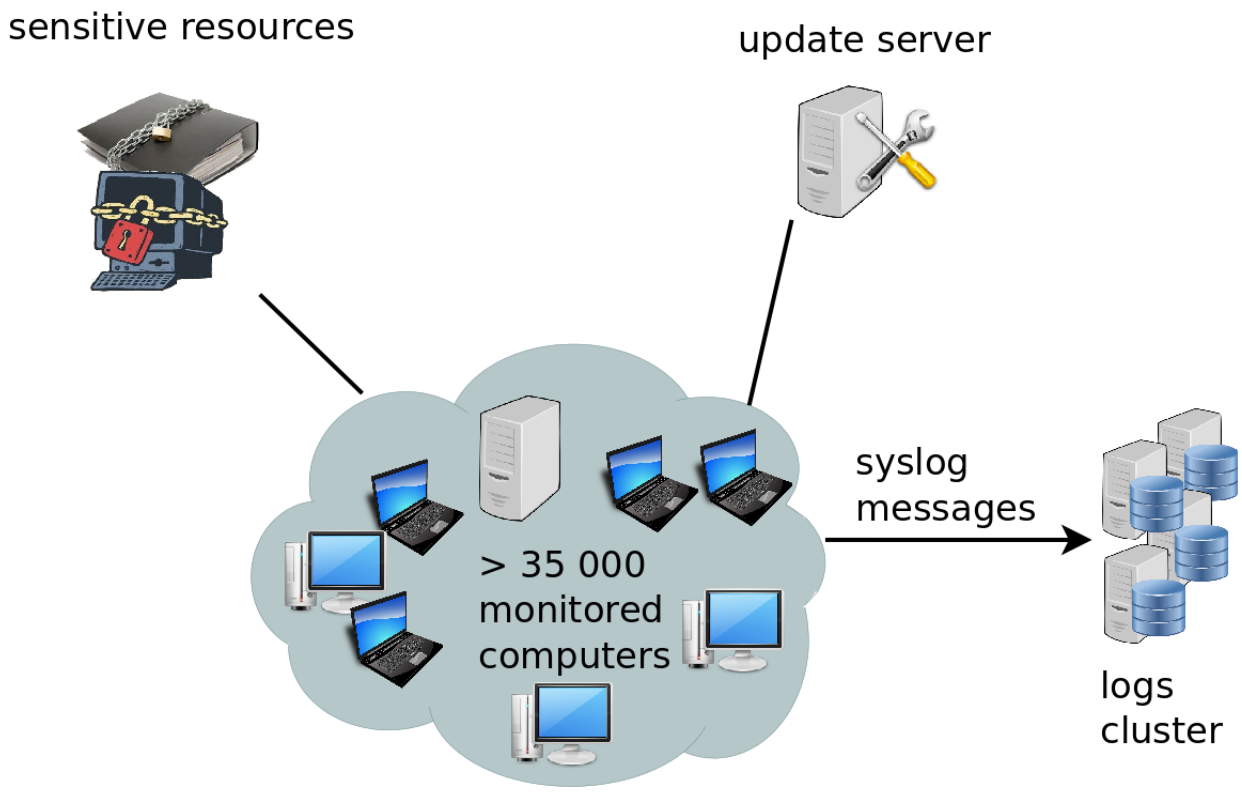
Algorithmic Realization with MapReduce

- MapReduce framework supports data-intensive distributed computations



- Advantages:
  - Computation distribution
  - Load-balancing
  - Fault-tolerance

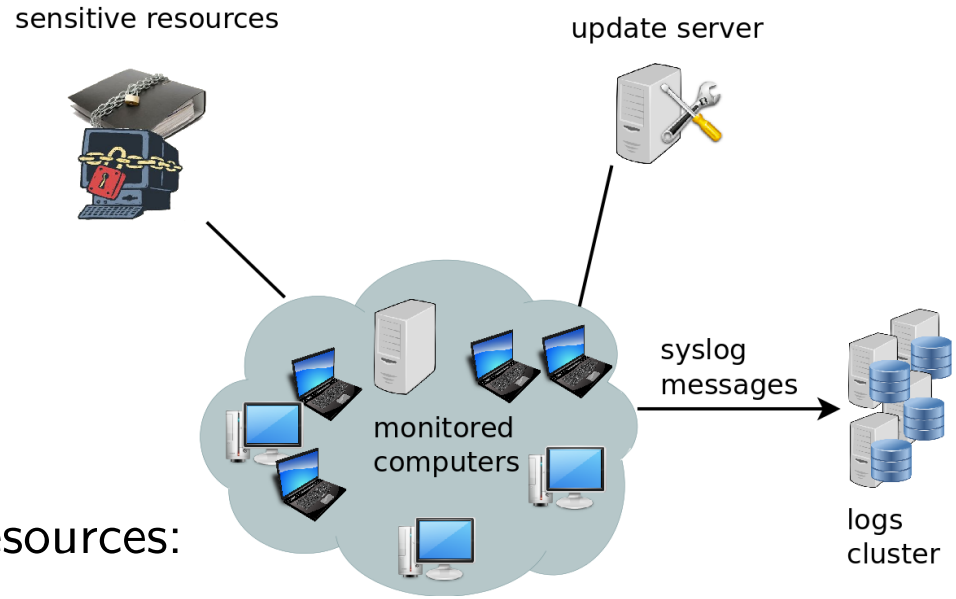
Google Case Study



- Verify hypotheses:
  1. Approach scales to very large logs.
  2. Slicers suitable for real-world policies.

- Logged 1 TB per day.
- Extracted log events:
  - Cover approximately 2 years.
  - 26 billion log events
  - 0.4 TB

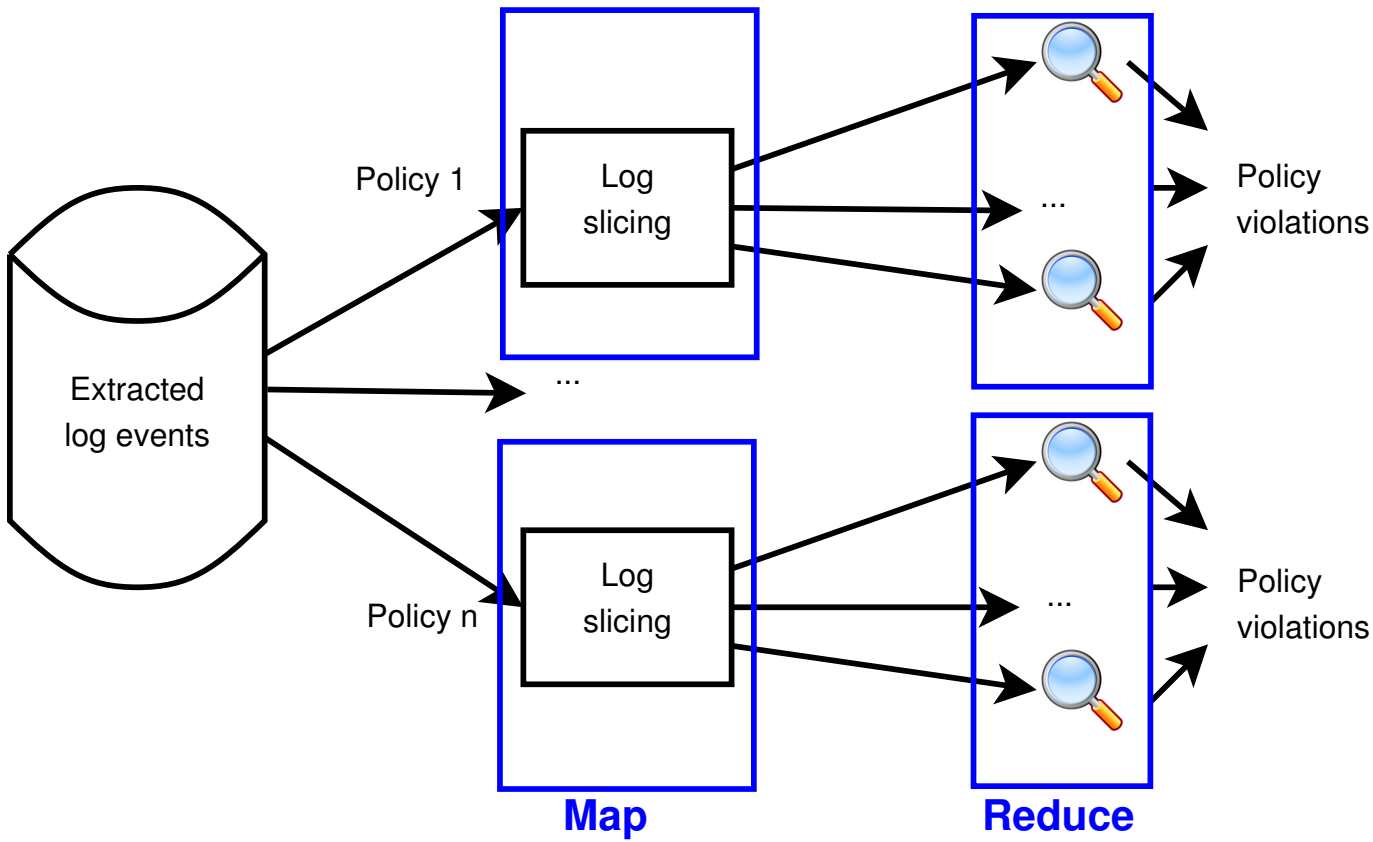
Policies



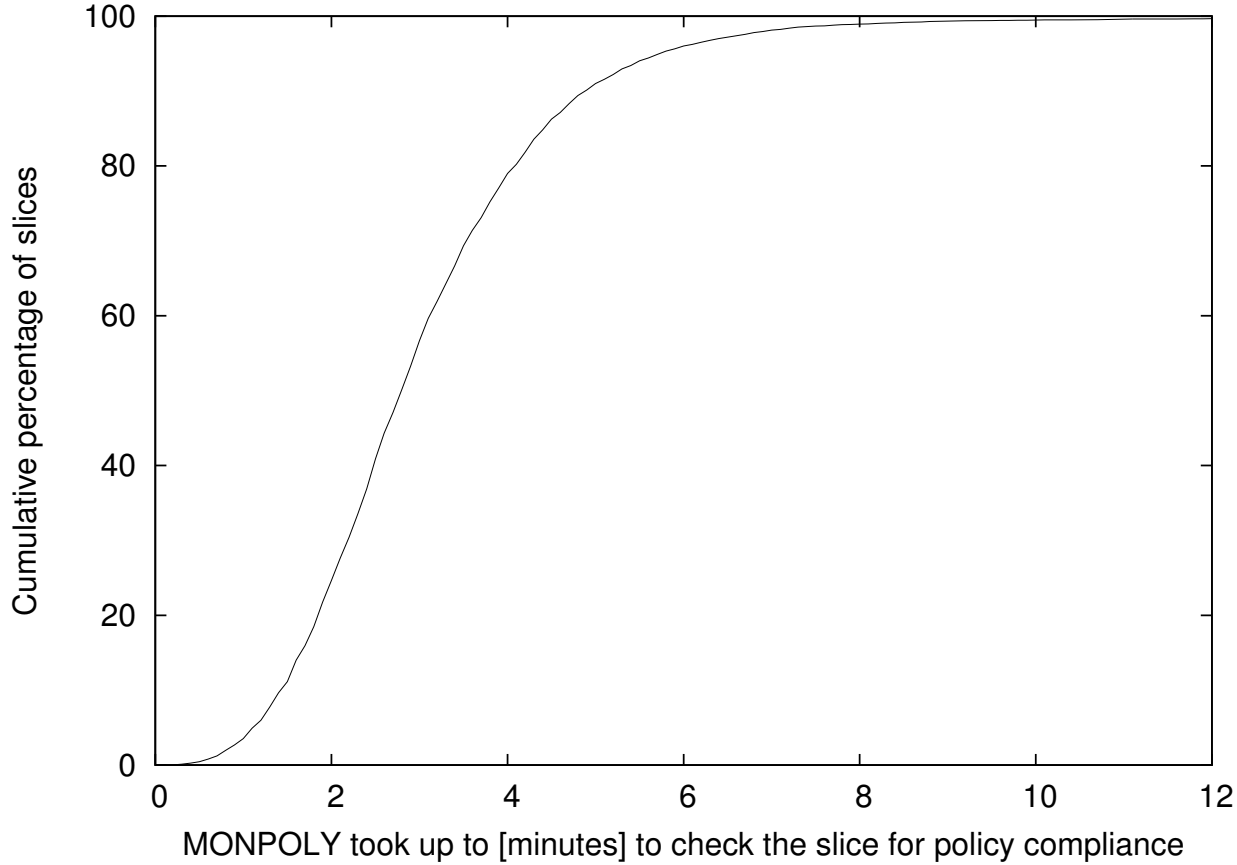
1. Authentication to access sensitive resources:
   - *P1*: Entering credentials takes at least 1 second.
   - *P2*: Only from updated computers (updated within the last 3 days).
2. SSH sessions:
   - *P3*: Not longer than 24 hours.
3. System updates:
   - *P4*: Update at least every 3 days.
   - *P5*: After downloading updates, apply them within 30 minutes.
   - *P6*: If a computer claims to be up-to-date, it must have updated during the last 24 hours.

Evaluation

- Log split into 10,000 data slices (by computer logging the event).
- For each policy, slices monitored by 1000 computers in parallel.
- Monitoring finished in 2.5 hrs, except for policy *P3* (12 hrs).

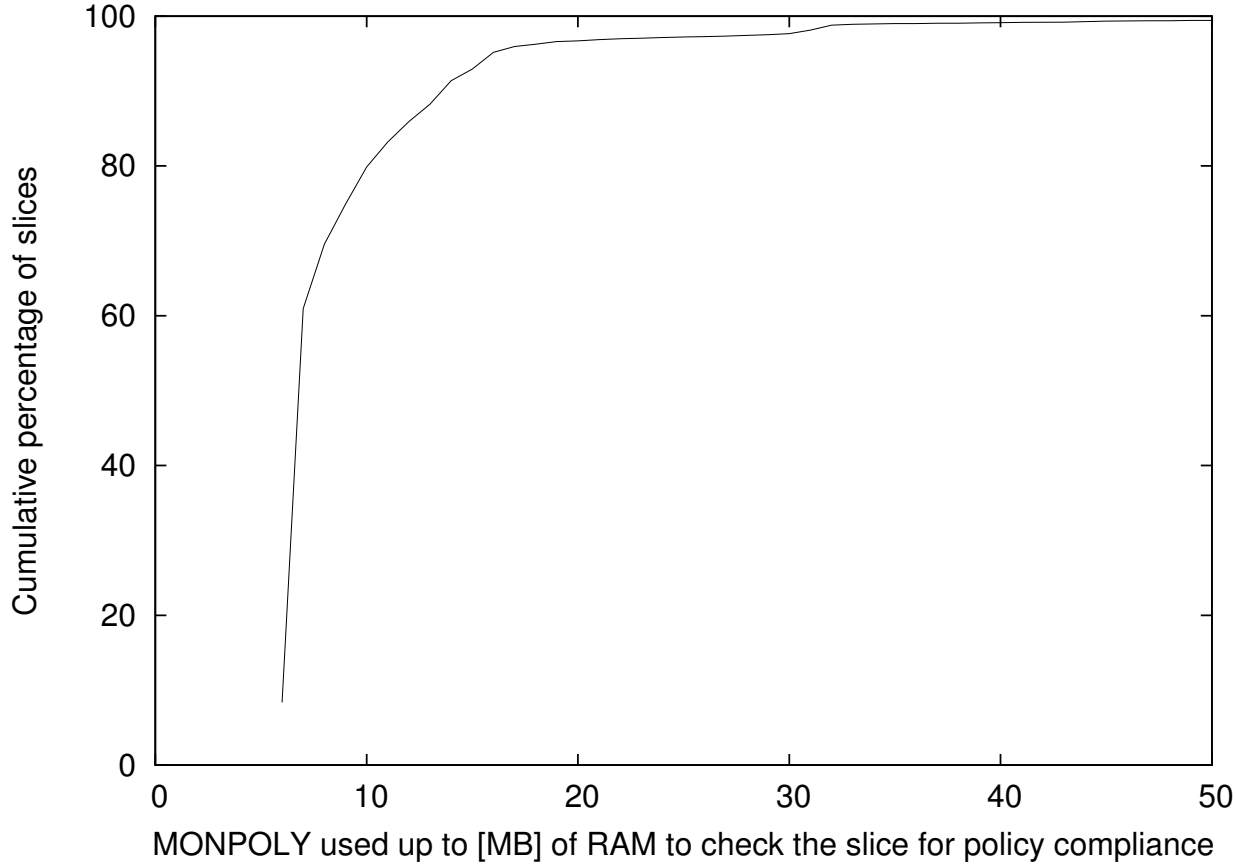


Time to Check a Single Slice



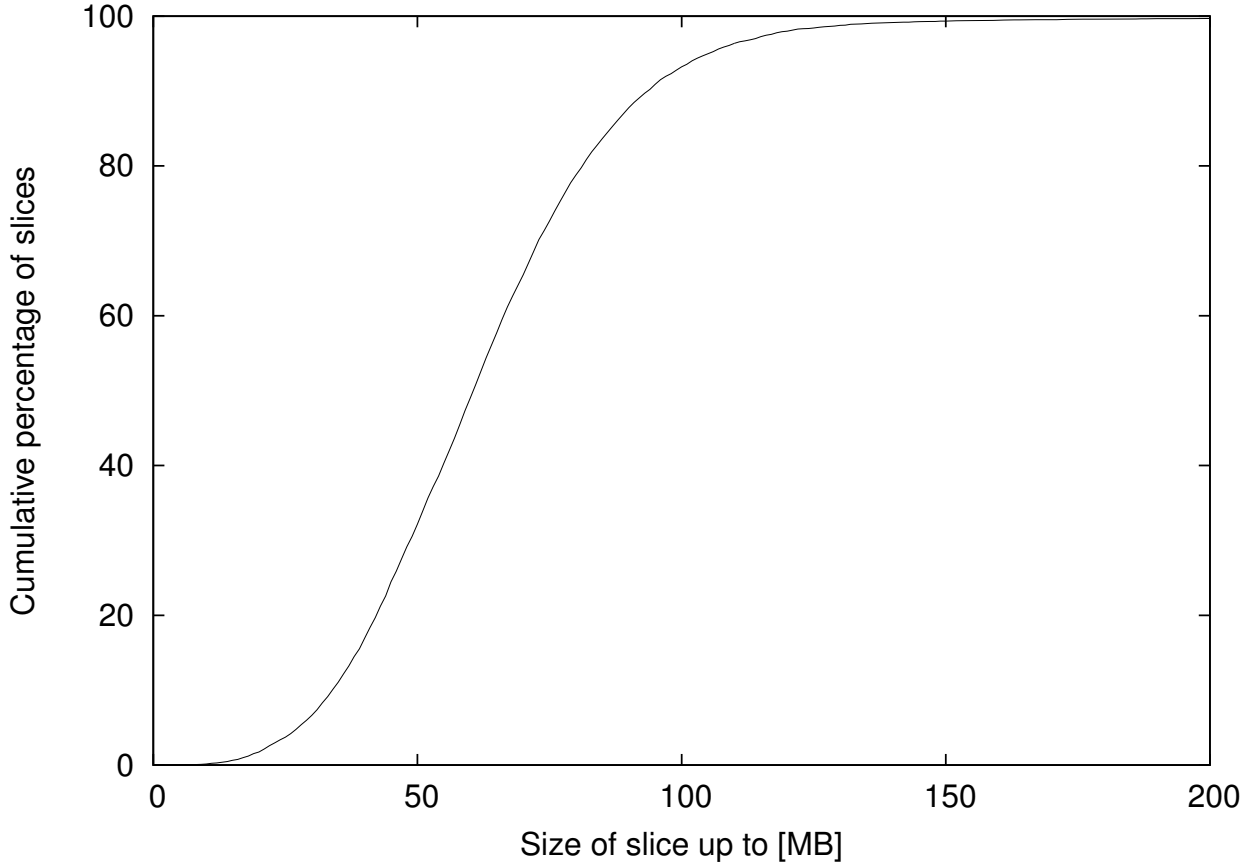
- 90% of slices monitored within 5 min.
- Median time: 3 min.
- Max time: 11 hours (policy *P3*).

Memory to Check a Single Slice



- 90% of slices monitored with less than 14 MB of memory.
- Median memory: 6–10 MB.
- Max memory: 510 MB (policy *P3*).

Distribution of Slice Sizes



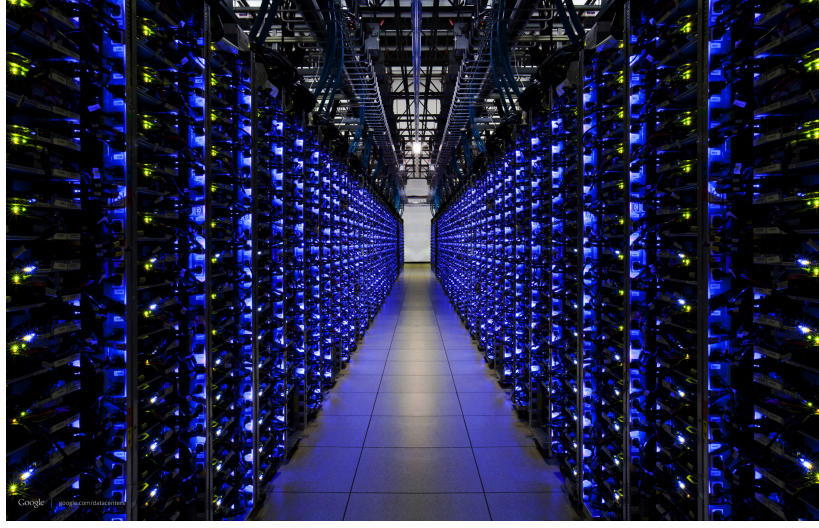
- 90% of slices smaller than 94 MB.
- Median size of a slice: 61 MB.
- Three slices over 1 GB, max 1.8 GB.

Related Work

- B. Barre, M. Klein, M. Soucy-Boivin, P.-A. Ollivier, and S. Hallé.
  MapReduce for parallel trace validation of LTL properties.
  RV 2012.
  - Parallelization based on formula structure rather than data.
    $\rightarrow$ Limited scalablity
  - Case study: $<$ 5 million log events, monitored on a single computer.

- G. Rosu and F. Chen.
  Semantics and algorithms for parametric monitoring.
  Log. Method. Comput. Sci., 2012.
  - Parametric temporal logic less expressive than first-order temporal logic.
  - Case study: 155 million log events, monitored on a single computer.
  - No distributed computation.

Conclusions



- **Scalable monitoring solution**
  - Framework to parallelize monitoring.
  - Algorithmic realization.
  - Case study.
- Very large case study (largest to date?) of applying runtime verification techniques to big data.
- **Future work**
  - Overcome offline limitation: use a stream processing platform.
  - Overcome unbalanced distribution of log slices.
  - Design and evaluate other slicing operators.

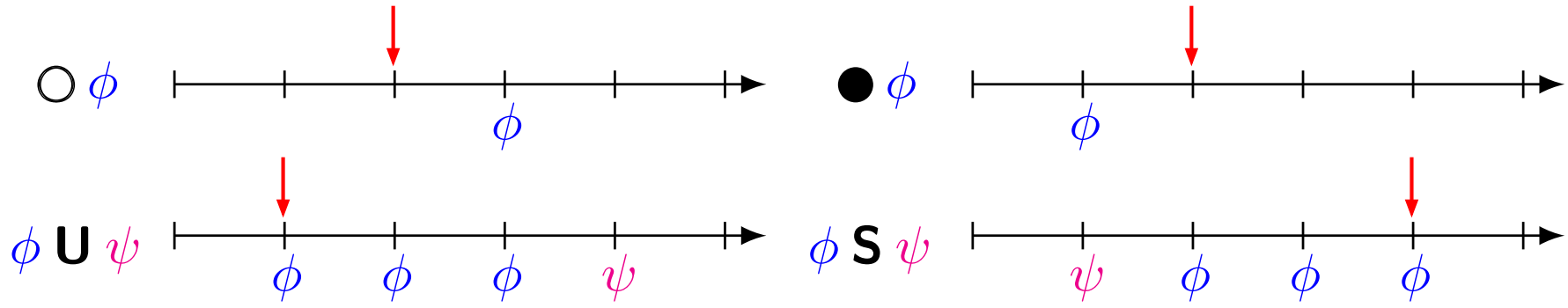
Backup slides

Metric First-Order Temporal Logic

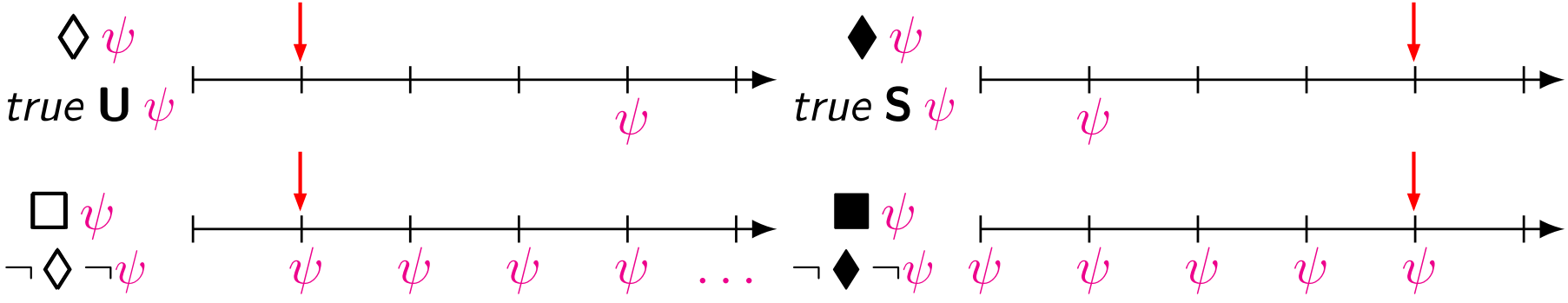
- First-order for expressing relations on data
  - $<$, $=$
  - $\neg$, $\wedge$, $\vee$
  - $\forall$, $\exists$
- Metric temporal operators for expressing qualitative and quantitative timing information
  - Past: $\bullet$, **S**, $\blacklozenge$, $\blacksquare$
    (`PREVIOUS`, `SINCE`, `ONCE`, `ALWAYSPAST`)
  - Future: $\bigcirc$, **U**, $\Diamond$, $\Box$
    (`NEXT`, `UNTIL`, `EVENTUALLY`, `ALWAYS`)
  - Metric operators add timing constraints,
    e.g., $\blacklozenge_{\leq 10}$ for once within 10 time units
    (`ONCE [0,11)`)

Standard linear temporal operators

- Primitive temporal operators

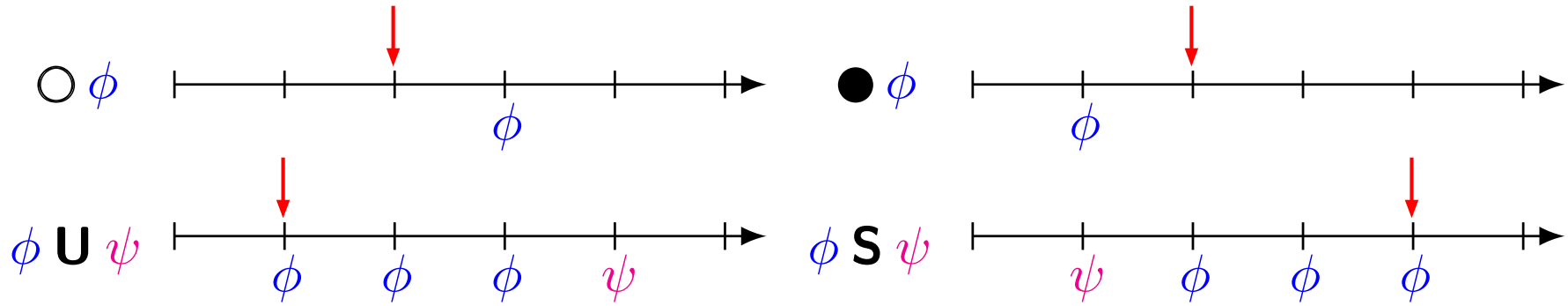


- Derived temporal operators

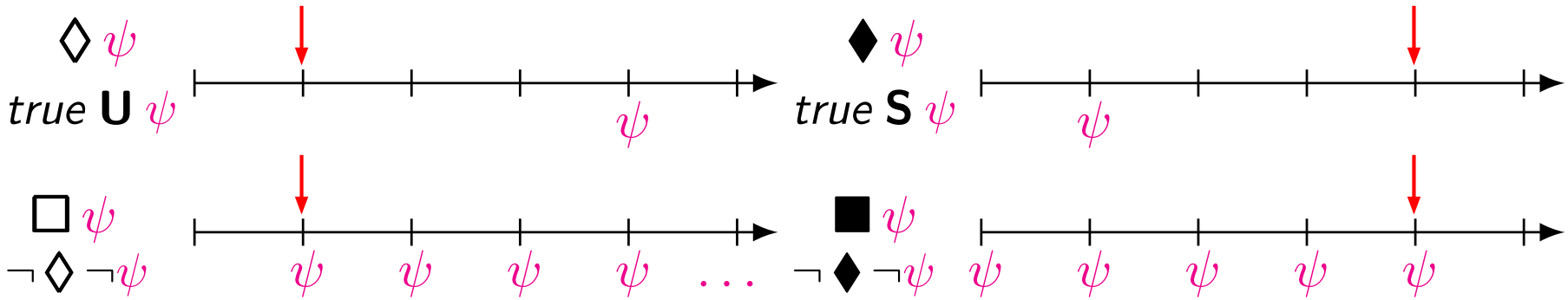


Metric temporal operators

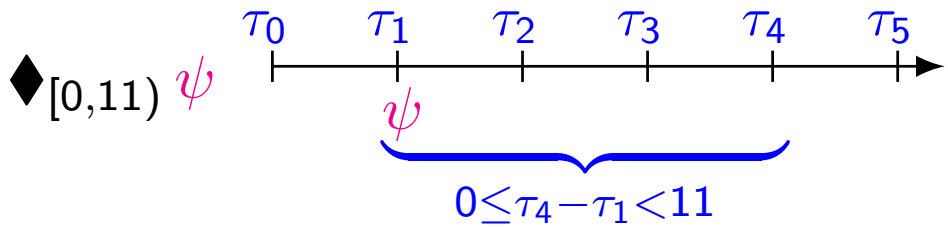
- Primitive temporal operators



- Derived temporal operators



- **Metric operators add timing constraints**



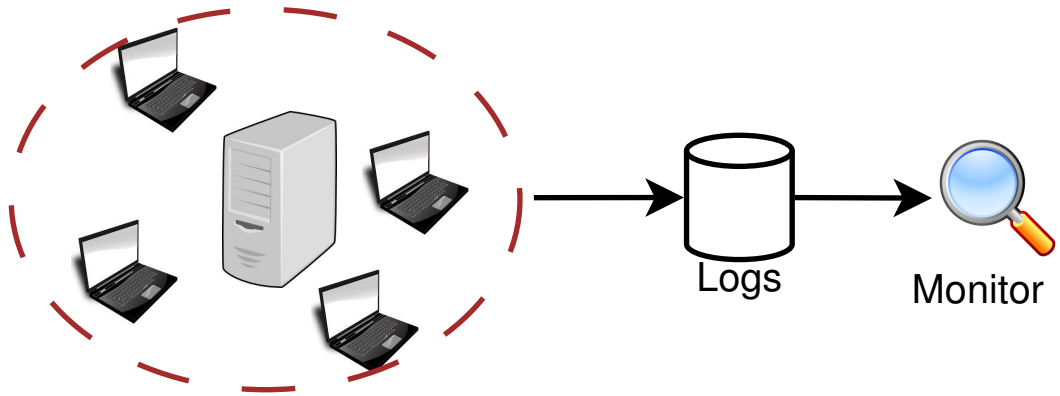
Monpoly

- Monitoring tool to determine
  which events in a log violate given rules.
  - Input: policy, log
  - Output: violations
- Policies specified in metric first-order temporal logic.
- Result of multi-year project at ETH Zurich.

Monpoly

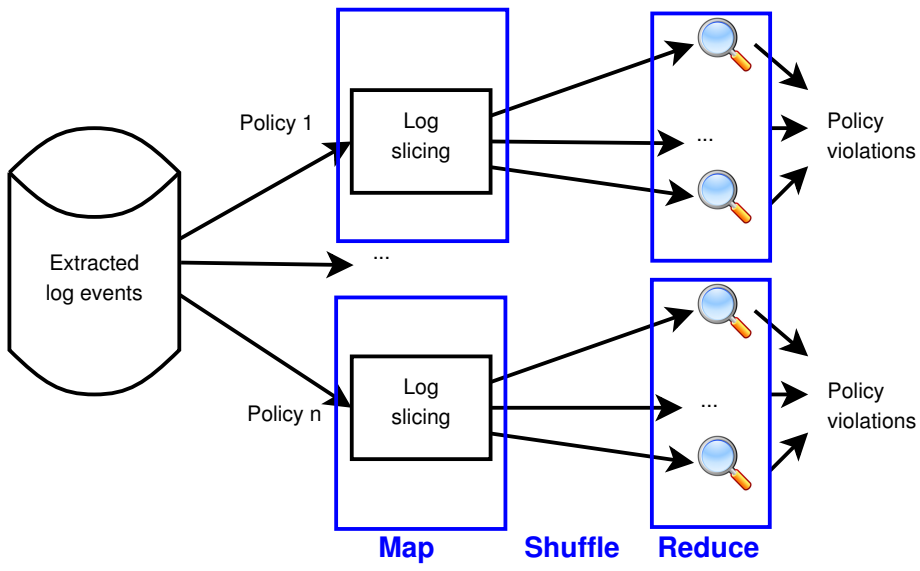
- Monitoring tool to determine
  which events in a log violate given rules.
  - Input: policy, log
  - Output: violations
- Policies specified in metric first-order temporal logic.
- Result of multi-year project at ETH Zurich.

- Single-threaded, not parallelized.



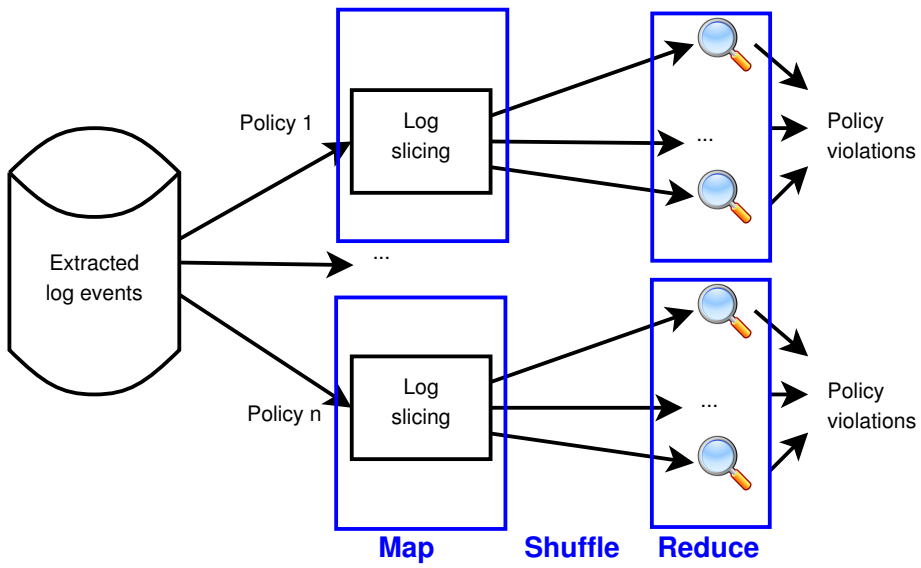
MapReduce Details

- Map: Split log into slices, emitting each log event with:
  - Primary key: log slice identifier
  - Secondary key: timestamp



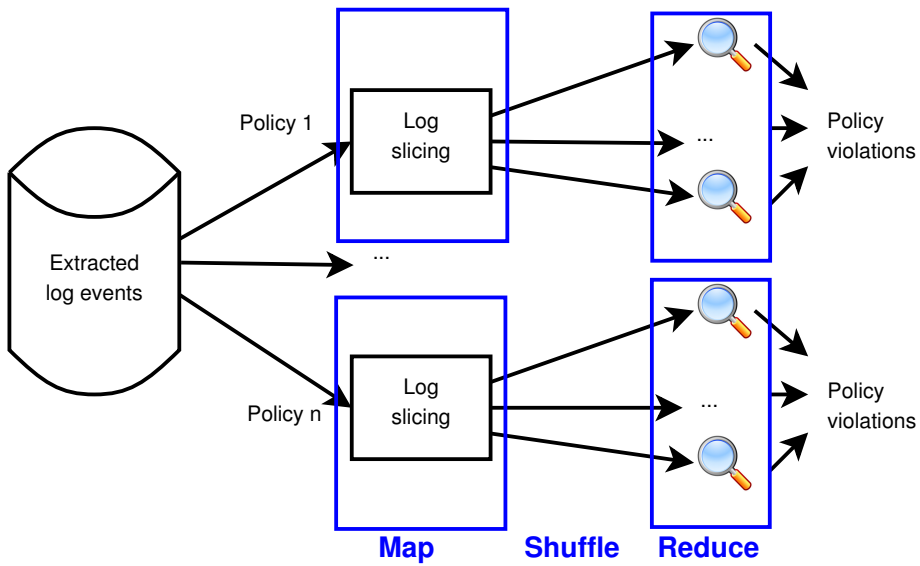
MapReduce Details

- Map: Split log into slices, emitting each log event with:
  - Primary key: log slice identifier
  - Secondary key: timestamp
- Shuffle: Collect all parts of a log slice.



MapReduce Details

- Map: Split log into slices, emitting each log event with:
  - Primary key: log slice identifier
  - Secondary key: timestamp
- Shuffle: Collect all parts of a log slice.
- Reduce: Monitor a log slice.
  - Monpoly runs in a child process.
  - Log slice piped into Monpoly.
  - Monpoly's output (policy violations) returned by reducer.



Logs

- Raw logs: 1TB of syslog messages per day
- Extracted log events:
  - Cover approximately 2 years.
  - 26 billion log events
  - 0.4 TB