Reading, Analyzing, and Presenting Scientific Papers

IN5420/9420, SPRING 2022

Classifying the type of contribution

- A tutorial or a textbook
- Elegant formalization of a problem that captures reality
- New or improved solution to an important and interesting problem
- New generic technique for analysis/solution/evaluation
- New analysis
- Better evaluation
- Survey

Important questions to consider

- Target application domain
  - What makes the work so suitable for this type of applications?
  - How general or easy is it to extend?
- Is the solution practical?
  - Example: the solution requires system engineer to define an exponential number of...
- Is the evaluation comprehensive and adequate?
  - Example: dependable system in failure-free settings or adaptive system with most parameters fixed

Important questions to consider (cont´d)

- What is the motivation for this work?
  - The authors certainly claim that it is important. What makes you believe it?
- Does the contribution and results correspond to the motivation? Do all pieces fit together?
- Where did the work fall short of your wild imagination? What extension would you like to see the most?
- Asking questions is always an option
  - Teachers, supervisors, colleagues
  - Even if you do a presentation alone

Guidelines for reading papers

- Read the paper in three passes
  - Pass I: the abstract, section titles, techniques, and references
  - Pass II: read the paper but prioritize, skip the proofs and elements less important for understanding the contribution
  - Pass III: try to re-implement the paper in your mind
    - Identify hidden assumptions
    - Identify weak and strong points
    - Missing comparison with related work
    - Potential issues with the experimental or analytical techniques
  - Refer to “How to read a paper” https://www.albany.edu/spatial/WebsiteFiles/ResearchAdvices/how-to-read-a-paper.pdf

Guidelines for student presentations

- Present authors’ claim regarding the work
  - Problem studied
  - Contributions of the paper
  - Relation to the state of the art
  - Depending on type of contribution and research method
    - Definitions, Requirements, Design, Algorithms, Implementations, Experiments, Evaluation, etc
  - Main results
  - Conclusion
  - Refer to http://www.stanford.edu/~jacksonm/present.pdf

Guidelines for student presentations (cont´d)

- Criticism
  - Asking and concluding on the important questions
  - Guidelines: “The Task of the Referee” by Alan J Smith, UCB
  - Guidelines: Section 4 in “Writing Reviews for Systems Conferences” by T. Roscoe
- Questions for discussion!
- Summary and conclusion of criticism

For presentations you will be better off skipping

- Formal parts
  - Unless it is a work on new proof techniques or formal methods
  - Formal notation and definitions
    - important for written papers
    - replace with intuition and examples
  - Theorem proofs and mathematical derivations
    - focus on the result and intuition behind the proof instead

For presentations you will be better off skipping (cont´d)

- Lengthy lists of related work
  - Instead present categories
  - Or one representative for each
- Fine-grain implementation details
  - Present the central part or implementation idea
    - Skip optimizations
  - Provide overview of pseudo-codes
    - and if require, present a few central lines
- Overly detailed architectural schemes
  - Papers typically include such schemes with dozens of entities and arrows, and annotations in a tiny font

Guidelines for preparing questions

Avoid following questions:

- Questions with answer well-known and easily found in the literature
  - E.g., the average time between blocks in Bitcoin
- Questions absolutely unrelated to technology
  - E.g., velocity of a given cryptocurrency
  - It is ok, however, to talk about the use and popularity of technology because it gives motivation for the technology
- Questions that we can have no hope to reason about
  - E.g., will cryptocurrencies be in use in 100 years

Good questions:

- Understanding of nontrivial points in the topic
- Technological outlook for the near future
- Comparison between studied technologies

Guidelines for writing reviews

- Hypothesis and Context
  - brief problem description
  - contribution
  - summarize the central ideas, research methodology and supporting points
- Novelty of the research
  - a new perspective on the topic,
  - applicable to real-world situations,
  - comparison with other approaches
  - providing concrete examples of where the approach succeeds or fails.
- Assessment
  - your opinion of the research

Walk-through a sample review

- Research Paper: "Minimizing Churn in Distributed Systems"
- Public review available at Sigcomm 2006

Motivation/Importance of the Problem

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on the perceived quality of service.

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce the churn rate and improve service. The question is: how do we find these nodes?

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes the churn incurred by some classes of strategies, and embodies a surprising result with practical implications.

The paper categorizes strategies for selecting nodes along two axes. *Predictive* strategies base their selection on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world traces that span peer-to-peer networks, corporate workstations, and public web servers.

The result is quite surprising: random replacement, despite its lack of smarts, performs better than preference list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the problems of detecting and properly classifying node outages caused by network failures.

The paper then examines the implications of this analysis on system design in the context of an anycast service, a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

Definition of the Problem

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce the churn rate and improve service. The question is: how do we find these nodes?

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes the churn incurred by some classes of strategies, and embodies a surprising result with practical implications.

The paper categorizes strategies for selecting nodes along two axes. *Predictive* strategies base their selection on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world traces that span peer-to-peer networks, corporate workstations, and public web servers.

The result is quite surprising: random replacement, despite its lack of smarts, performs better than preference list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the problems of detecting and properly classifying node outages caused by network failures.

The paper then examines the implications of this analysis on system design in the context of an anycast service, a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

1

Overview of Contributions

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on the perceived quality of service.

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes the churn incurred by some classes of strategies, and embodies a surprising result with practical implications.

on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world traces that span peer-to-peer networks, corporate workstations, and public web servers.

The result is quite surprising: random replacement, despite its lack of smarts, performs better than preference list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the problems of detecting and properly classifying node outages caused by network failures.

The paper then examines the implications of this analysis on system design in the context of an anycast service, a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

Research Methodology

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on the perceived quality of service.

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce the churn rate and improve service. The question is: how do we find these nodes?

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes

The paper categorizes strategies for selecting nodes along two axes. *Predictive* strategies base their selection on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world traces that span peer-to-peer networks, corporate workstations, and public web servers.

list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the problems of detecting and properly classifying node outages caused by network failures.

The paper then examines the implications of this analysis on system design in the context of an anycast service, a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

1

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on the perceived quality of service.

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce the churn rate and improve service. The question is: how do we find these nodes?

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes the churn incurred by some classes of strategies, and embodies a surprising result with practical implications.

The paper categorizes strategies for selecting nodes along two axes. *Predictive* strategies base their selection on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world

Core Result

The result is quite surprising: random replacement, despite its lack of smarts, performs better than preference list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the problems of detecting and properly classifying node outages caused by network failures.

a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

Minimizing Churn in Distributed Systems

P. Brighten Godfrey Scott Shenker Ion Stoica

Public Review by Emin Gün Sirer

Churn, caused by the departure and arrival of nodes, is the bane of modern distributed systems. At best, churn necessitates additional bandwidth and coordination to transfer service functionality from failed nodes to others. Often, it leads to service interruption, with user-visible side-effects. And there are many existing distributed services where the churn rate, if over a given threshold, would overtake the system's ability to recover and thus render the service completely inoperable.

And it's not just the large-scale systems that are affected by the adverse effects of churn. To be sure, peer-to-peer systems need to pay careful attention, since their sheer size can exacerbate the overheads associated with even modest churn rates. But churn is also an issue for small systems, where making a bad choice can have a significant impact on the perceived quality of service.

Clearly, a principled approach to minimize churn, one that might subsume the various heuristics that have been proposed in past literature, is called for. Luckily, churn can be managed through the judicious selection of nodes which are available for admission into the system; selecting the nodes that are likely to remain up for a long time will reduce the churn rate and improve service. The question is: how do we find these nodes?

The current paper presents a quantitative guide to churn resulting from different node selection strategies, analyzes the churn incurred by some classes of strategies, and embodies a surprising result with practical implications.

The paper categorizes strategies for selecting nodes along two axes. *Predictive* strategies base their selection on node characteristics such as past uptime, whereas *agnostic* strategies do not take such measures into account. *Replacement* strategies perform dynamic replacement of failed nodes, whereas *fixed* strategies select the nodes they will use prior to deployment and stick to that set for their entire operation.

Three points among the spectrum of node selection strategies examined by this paper will resonate with practitioners. A predictive replacement strategy, embodied in many systems, is to prefer nodes with the longest current uptime. Another approach, an agnostic replacement strategy, is to select nodes according to a preference list based on a measure, like a hash, that does not intend to optimize churn. A final approach, also an agnostic replacement strategy, is to simply select replacement nodes at random.

The paper provides a precise definition of churn and quantitatively examines the churn incurred by various strategies. It examines the performance of different strategies under a synthetic trace as well as an extensive set of real-world traces that span peer-to-peer networks, corporate workstations, and public web servers.

The result is quite surprising: random replacement, despite its lack of smarts, performs better than preference list strategies, and is quite close to the best strategy under most scenarios. The paper then analytically models the performance of the random replacement strategy, and shows that it approximates selection based on longest uptime. There are, however, a few key differences with longest uptime that make random replacement more compelling: (i) it is very simple to implement (ii) it does not rely on truthful reporting of uptime by a peer, and (iii) it avoids the

Evaluation

The paper then examines the implications of this analysis on system design in the context of an anycast service, a DHT-based lookup service, a multicast tree maintenance algorithm and two replica placement strategies for DHT-

1

Evaluation

based storage services. The last scenario extends the quantitative analysis to a setting where nodes carry long-term state. For each scenario, the paper examines end-to-end performance metrics, and discusses how randomization can be used to transform commonly used strategies that, in essence, equate to a preference list into node selection criteria that reduces churn by approximating random selection. The changes required are straightforward and small, yet have an appreciable impact on the performance of the systems.

more frequently than others; an issue the paper defers to future work.

It is rarely the case that system designers can be told to do less to improve performance. This paper shows that less is, in this context, better.

Open Issues/Problems

based storage services. The last scenario extends the quantitative analysis to a setting where nodes carry long-term state. For each scenario, the paper examines end-to-end performance metrics, and discusses how randomization can be used to transform commonly used strategies that, in essence, equate to a preference list into node selection criteria that reduces churn by approximating random selection. The changes required are straightforward and small, yet have

An open issue in the paper is the impact of these selection criteria on fairness. Some nodes may be utilized much more frequently than others; an issue the paper defers to future work.

is, in this context, better.

2

Take-away Message

based storage services. The last scenario extends the quantitative analysis to a setting where nodes carry long-term state. For each scenario, the paper examines end-to-end performance metrics, and discusses how randomization can be used to transform commonly used strategies that, in essence, equate to a preference list into node selection criteria that reduces churn by approximating random selection. The changes required are straightforward and small, yet have an appreciable impact on the performance of the systems.

An open issue in the paper is the impact of these selection criteria on fairness. Some nodes may be utilized much

It is rarely the case that system designers can be told to do less to improve performance. This paper shows that less is, in this context, better.

2

Reading material

- A general collection of links on https://www.cs.columbia.edu/~hgs/etc/writing-style.html
- Reading and Presenting Research Papers, http://www.cs.rpi.edu/academics/courses/spring99/robotics/paperdiss.html
- Review
  - ACM Computing Reviews guidelines, http://www.computingreviews.com/Reviewer/RVguidelines2015.pdf
  - “The Task of the Referee” by Alan Jay Smith, especially the “Evaluating a research paper” section
- Presentation tips
  - http://www.stanford.edu/~jacksonm/present.pdf
  - http://ksuweb.kennesaw.edu/faculty/rbrow211/papers/presenting-research.html