HOW TO
do good systems research and publish papers

Changwoo Min
ECE @ Virginia Tech
Oct 23rd, 2018

Executive Summary

- No magic wand
  - Read papers; think critically; build systems; measure performance; write papers
- But there are anti-patterns and best practices
- Following are some thoughts and suggestions that I learned from my advisers, collaborators, and colleagues.

Three most important things in doing systems research

- Problem selection
- Problem selection
- Problem selection

How to find a good problem

- What is a good problem?
  - Important problem
  - Unsolved problem (or problem that needs more solutions)
  - New problem
- What should I do?
  - Read papers; think critically; build systems; measure performance; write papers
- Where should I take a look?
  - Emerging new technologies (manycore, accelerator, RDMA, non-volatile memory)
  - Boundaries between layers (SSD vs. file system)
  - New aspects of a system (security, energy consumption)
  - New applications (IoT, Mobile, Cloud)
  - Interactions between subfields (OS, concurrency, database)

Why finding a good problem is not easy as a gradstudent?

- **You should change your role in learning**
  - Undergrad: knowledge ***consumer***
  - Grad: knowledge ***producer***
- **Are you ready to find an important problem?**
  - Student: I have a cool idea, which is …
  - Adviser: Sounds great! The similar idea was published at 197x by someone who won the Turing award.

    / a month later */*
  - Student: I have another cool idea, which is …
  - Adviser: Sounds great! The similar idea was published at 198x by someone who won the Turing award.

    / a month later */*

    … 199x …

    / a month later */*

    … 200x …`

Develop a good intuition

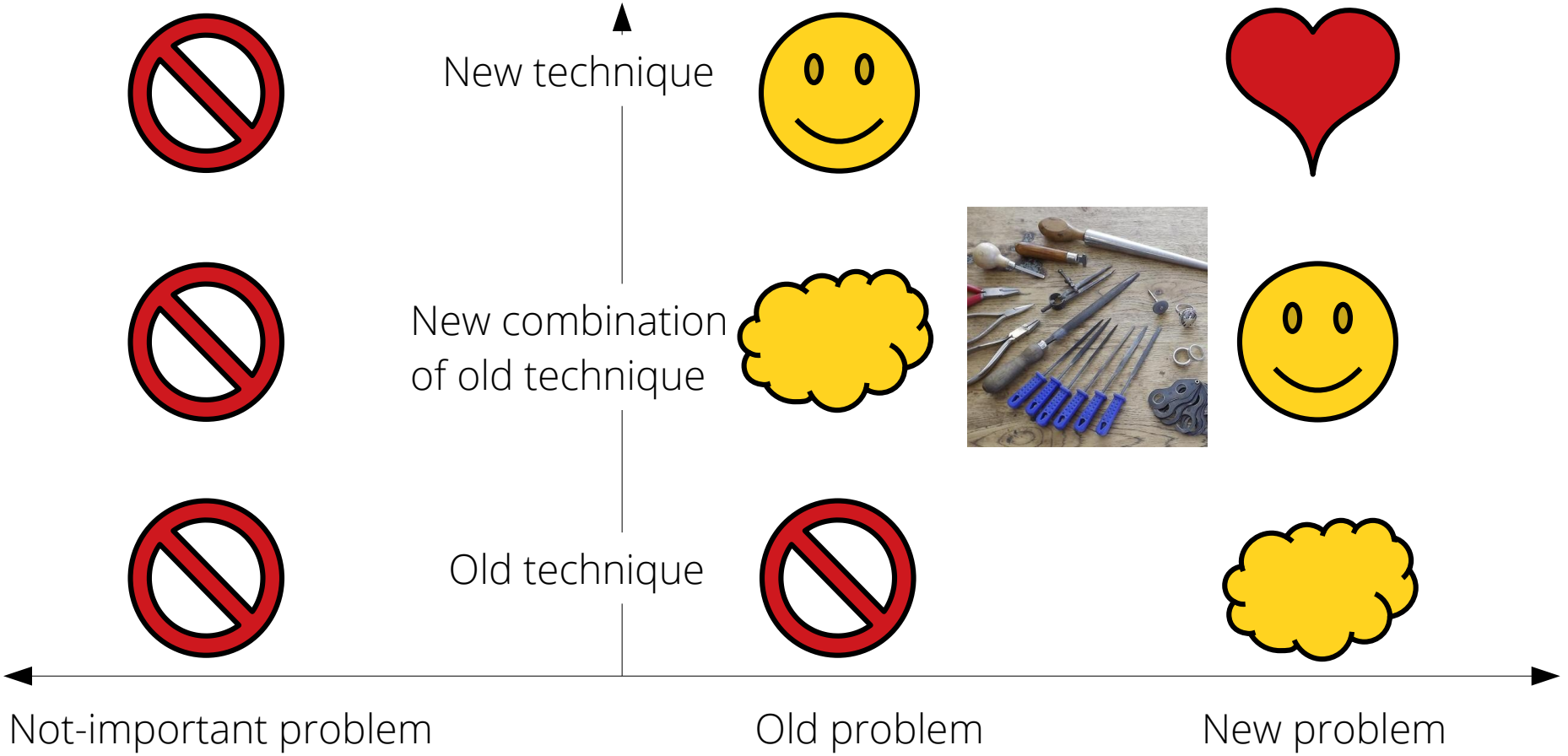
- How to approach the problem
  - What's the *fundamental* limitation of existing approaches?
  - Why do you think your approach would be better?
  - E.g., MV-RLU: two threads accessing different memory locations should run in parallel.
- Elevator pitch to your adviser, colleagues, senior meet in conferences
  - What are their first questions?
  - Why they are not enthusiastic to your idea?
- Improve your intuition/idea/algorithm
- Repeat
- Do you still think your approach is good?

Is your idea publishable? (1/2)

- Is your system better?
  - Better performance
  - Short tail latency
  - Save energy
  - Enable new types of applications
  - Secure by design
  - Etc.
- How much better is “better”?
  - 100x, 10x
  - 30%
  - ~~10%, 5%~~

Is your idea publishable? (2/2)

- Is your idea novel?



My rule of thumb to find an important problem and a good solution : 30-20 rule

- First read 30 papers
  - Understand the problem space more deeply
  - Know the details of common techniques and the big ideas behind them
  - Know how to evaluate and what to compare
  - Find out the fundamental limitations of the 30 papers
  - Come up with your tentative solution to overcome the fundamental limitations
- Then read another 20 papers
  - Understand techniques proposed in 20 more papers
  - Test if your idea is competitive over the 20 papers at least one or two aspects (e.g., performance, energy, programmability, etc)

Is it doable?

- Can you develop the first prototype within a semester?
  - If no, ...
    - Learn necessary skills
    - Find good team mates
    - Limit the scope → Is your idea still competitive?
- Know your competition
  - What is the baseline?
  - What is the state-of-the-art?

Design / Implementation / Evaluation

- This is an iterative process.
  - Start the minimal design and implementation
  - Gradually add features
  - Or periodically rewrite the code
- Plan evaluation ahead
  - Write unit testcases
  - Integrate your benchmark and automate evaluation

Papers in systems research

- Romantic comedies

Papers in security research

- Superhero

Superhero Movies Treat Wo...
mic.com

Superhero Movie (2008) - I...
imdb.com

Top 10 Superhero Movies!
thesefantasticworlds.com

41 best Superhero Movies im...
pinterest.com


Top 50 Rotten Tomatoes S...
indiewire.com

Iron Man (2008 film) - Wiki...
en.wikipedia.org

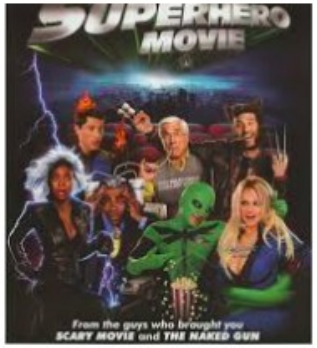
Superhero Movies Treat Wo...
mic.com


Ant-Man (film) | Marvel Mov...
marvel-movies.wikia.com

Movie Review: "Captain A...
pinterest.com

Top 21 Superhero Movies ...
saltypopcorn.com.au

Superhero Movie Poster | ...
movieposteraddict.com



movie poster warehouse mov...
movieposter.com

The Law of Genre

ARCHPLOT STRUCTURE

(AKA: Classic plot, the hero's journey, goal-oriented plot, Aristotelian story shape, energeia plot, and Hollywood screenwriting structure)

Archplot is a goal-oriented plot where, "for better or worse, an event throws a character's life out of balance, arousing in him the conscious and/or unconscious desire for that which he feels will restore balance, launching him on a Quest for his Object of Desire against forces of antagonism (inner, personal, extra-personal). He may or may not achieve it" (McKee, 196). **Film Examples:** *Toy Story, The Godfather, Back to the Future, Star Wars,* Etc. (Most American Hollywood films use arch plot). **Book Examples:** *Harry Potter* (Rowling), *Hunger Games* (Collins), *Speak* (Anderson), *Pride & Prejudice* (Austen), *Hamlet* (Shakespeare), *The Odyssey* (Homer), etc.

"I took a master class with Billy Wilder once and he said that in the first act of a story you put your character up in a tree and the second act you set the tree on fire and then in the third you get him down." - Gary Kurtz (*Film Producer*)

Ordinary World
Also known as:
- *The Known*
- *The Set-Up*
- *The Status Quo*
- *Limited Awareness*

Call to Adventure
Also known as:
- *Inciting Incident*
- *The Call to Action*
- *The Catalyst*

Refusal of the Call
Also known as:
- *Threshold Guardians*
- *Defining Moment*
- *Separation*
- *Reluctance*
- *New Situation*
- *The Debate*
- *Meeting Mentor*

Crossing the First Threshold
Also known as:
- *Energetic Marker 1: End of the Beginning*
- *The Point of No Return*
- *Committing to the Goal*
- *Act One Climax*
- *Plot Point One*
- *Break into Two*
- *Turning Point One*
- *The Threshold*
- *Awakening*

Tests, Allies, and Enemies
Also known as:
- *The Fun and Games*
- *Resistance and Struggle*
- *Rising Action and Obstacles*
- *Belly of the Whale*
- *Push to Breaking Point*
- *The Special World*
- *Road of Trials*

Mid-Point
Also known as:
- *Energetic Marker 2: Halfway Point*
- *Mid-Act Climax*
- *Moment of Grace or the Mind-Fuck Moment*
- *Moment of Enlightenment*
- *Commitment to the Journey*
- *Progress*

Approaching Inmost Cave
Also known as:
- *Challenges and Temptations*
- *Grace and Fall*
- *Resistance and Struggle*
- *Complications and Higher Stakes*
- *The Bad Guys Close In*
- *Intensification*
- *Preparation*
- *Rising Action*
- *Obstacles*

Inmost Cave
Also Known As:
- *Energetic Marker 3: Crisis*
- *Dark Night of the Soul*
- *Abyss and Revelation*
- *Plot Point Two*
- *Act Two Climax*
- *The Major Assault*
- *Death of the Ego*
- *Death Experience*
- *Rock Bottom*
- *The Ordeal*
- *The Crisis*
- *Big Change*
- *Epiphany*

Final Push
Also known as:
- *The Descent*
- *The Sprint*

Seizing the Sword
Also known as:
- *Energetic Marker 4: Climax*
- *The Climax*
- *Seizing the Prize*
- *Transformation*
- *Finale*
- *Break Into Three*
- *The Final Incident*

Return with the Elixir
Also known as:
- *Transformation and Return*
- *Rapidly Falling Action*
- *The Road Back*
- *Denouement*
- *New Life*
- *Resolution*
- *Aftermath*
- *A New Status Quo*

1 10 20 | 30 40 50 60 70 80 | 90 100 110 120

ACT ONE: BEGINNING
Set-Up / Incitement / Ordinary World

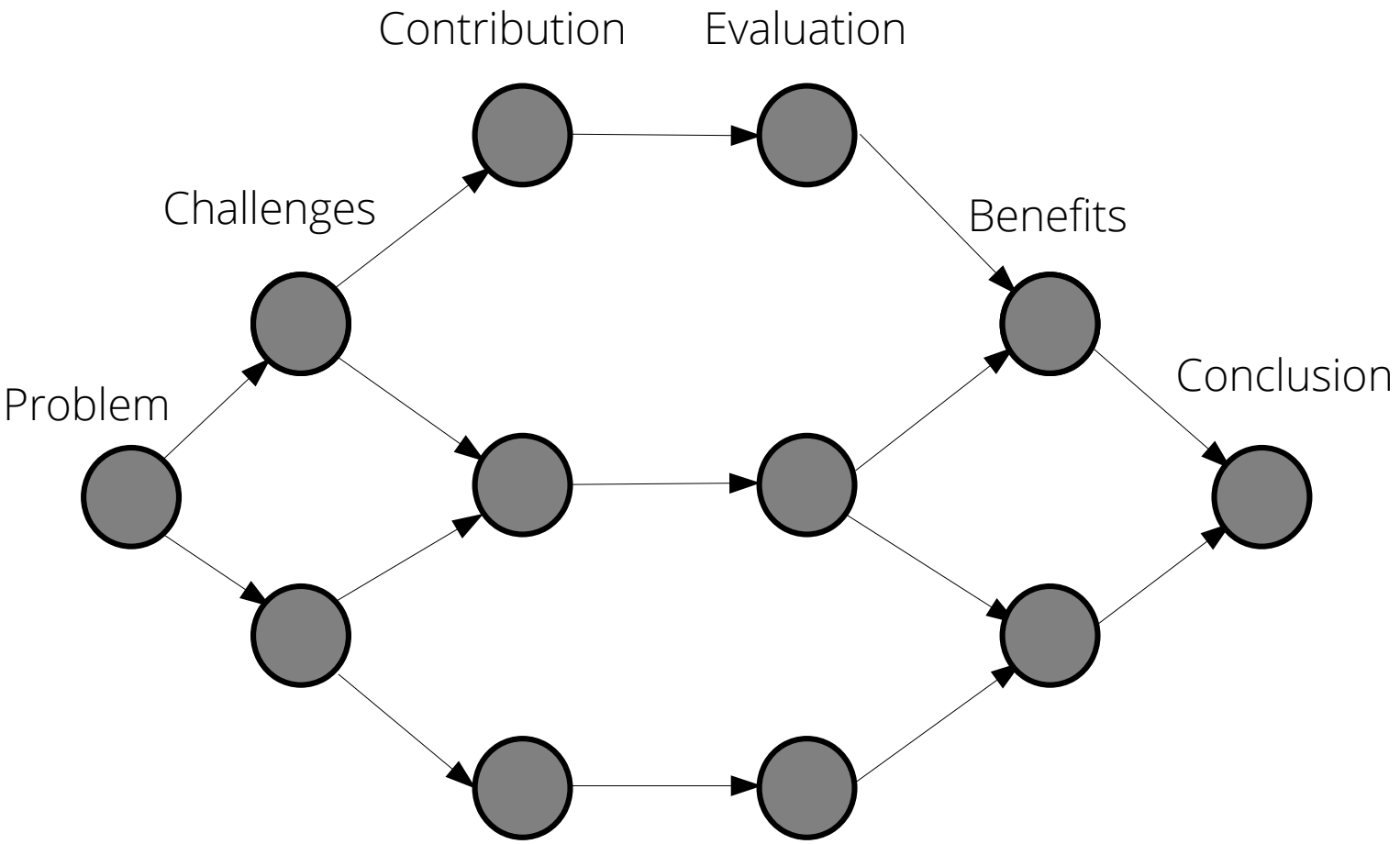
ACT TWO: MIDDLE
Confrontation / Set Back / Special World

ACT THREE: END
Resolution / Climax / Ordinary World

An average Hollywood film is 120 minutes long. Using screenplay format, one page = one minute of screen time.

Information on this handout comes from the following sources: *The Plot Whisperer* (Alderson), *The Hero with a Thousand Faces* (Campbell), *Novel Writing Help* (Chapman), *Screenplay* (Field), *The Sequence Approach* (Guilino), *Writing Screenplays that Sell* (Hauge), *Inside Story* (Marks), *Story* (McKee), *Tools for Analyzing Prose Fiction* (McManus), *Save the Cat* (Snyder), *Three Act Structure* (TV Tropes), *The Writer's Journey* (Vogler), and *The Moral Premise* (Williams).

Archplot of a systems paper



Before writing a paper

- Know your strength and weakness
  - Read all related papers: 30 – 50
  - What is the real strength of my approach?
  - How will you defend your weakness?
- Does your evaluation results support your claim?
  - Get baseline numbers first
  - Run core evaluation results
- Is your system better? Really?

Typical outline of a systems paper

- Introduction
- Background and Motivation
- Design
- Implementation
- Evaluation
- Discussion and Limitation
- Related work
- Conclusion

Outlining your paper

- Top-down approach
- Find 2-3 most similar paper in style
  - Copy the style of the paper!
- Quickly outline the paper (within one day)
  - Decide section title
  - For each section
    - Key message of each paragraph
  - Evaluation
    - What to show? What to compare?
- Present the first version of outline to co-authors
  - What's their questions?
- Rewrite the outline from scratch (withing five hours)
  - Don't try to modify the first version
- Revise the outline
  - By co-authors

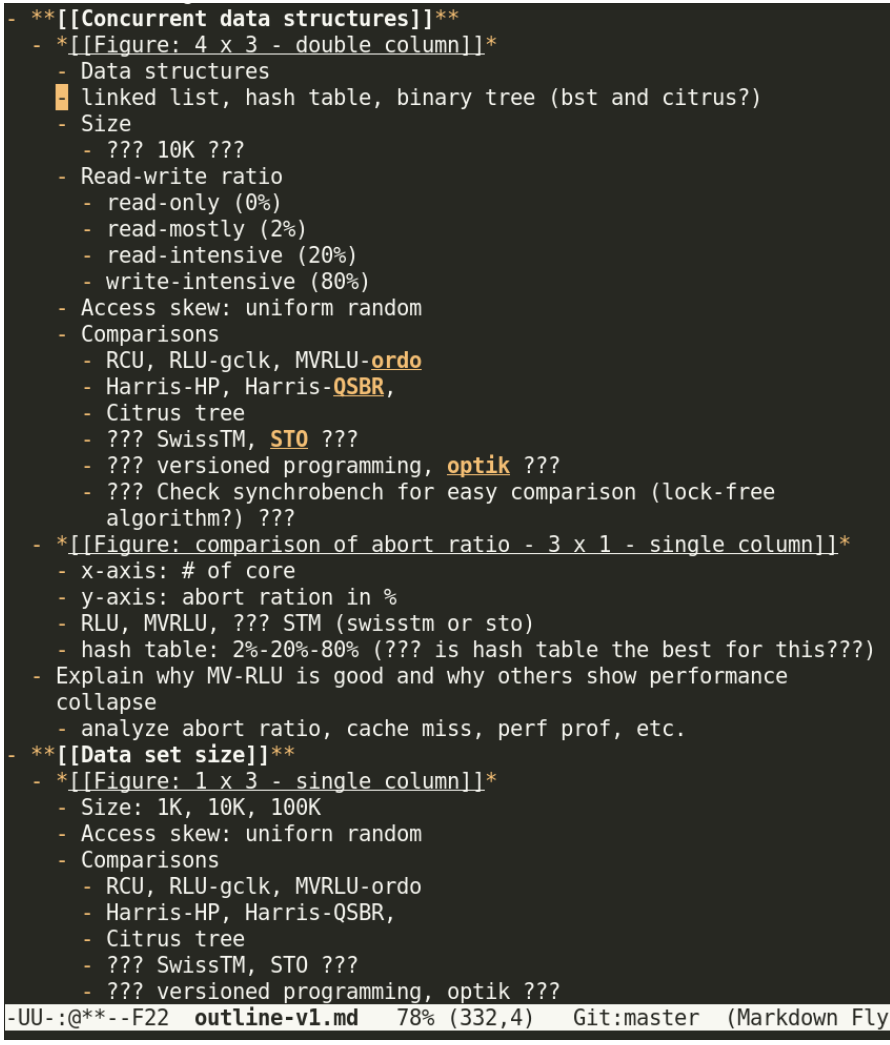
Outline examples

```
# MV-RLU: Scaling Read-Log-Update using Multi-Version Concurrency Control


## Introduction
- **[[Set context]]**
    - CDS is everywhere (e.g., kernel, database system, key-value store, DNS,
    - *[CDS is the key to scale many software in manycore systems.]*

- **[[Why this is important?]]**
    - More core -> small fraction of sequential execution -> performance coll
    - Amdhals' Limit: (e.g., 1% sequential part)
        - 10 cores  -> 9.2x
        - 32 cores  -> 24.4x
        - 64 cores  -> 39.2x
        - 128 cores -> 56.4x
        - 224 cores -> 69.3x
        - 448 cores -> 81.9x
    - Such big machines are real
        - 64 cores -> a typical server in a data center (ref: Facebook)
        - 192 cores (24 core * 4 smt * 2 socket) -> new arm
          server (ref: news article)
        - 448 cores (28 core * 2 smt * 8 socket) -> Intel Xeon server (ref: r
    - *[This is an increasingly important and interesting problem.]*


- **[[Introduce related work very briefly]]**
    - Significant research efforts since the first inception of
      multiprocessors (refer Table 1 and Fig 1)
    - Locking: mutual exclusion -> sequential execution
        - more parallelism while guaranteeing mutual exclusion
        - fine grained locking: linux VFS (XX -> YY)
        - more permissive lock:
            - inode mutex -> inode rswem: multiple readers
            - sequence lock in inode cache: multiple readers, single writer,
        - pros: intuitive, composible
        - cons: deadlock, livelock, complex, finer grained -> higher synch ov
    - Lock free algorithm: memory hotspot, high synch cost
        - Difficult to design, hard to correct, not composible
        - Difficult to scalee
        - Scalable memory reclamation scheme
-UU-:@**--F20  outline-v0.md   Top (1,0)     Git-master  (Markdown Fly Helm A
```

```
1. Intro
- Tech trends
- Exim
- Summary of our finding
- Summary of contrigution
- Paper organization

2. Case study
- Multi-threaded IO is popular.
  - desktop [file is not a file]
  - mobile [revisiting, quasio]
  - cloud [rocks db, wiredtiger, ...]
- How applications make an effort to parallelisze file system operations?
  - And their results on ramdisk/ssd/hdd
- Exim
  - message file creation at spool dir. and move the the user dir.
  - use three processors
- RocksDB
  - Write performance in write-optimized LSM-like key/value store
    is limited by compaction performance.
  - Parallel compaction
    merge two files into another two
- MariaDB/InnoDB
  - reader thread
  - writer thread
  - depending on operation (read/write) queue specific operation into separ
    foreground-thread can continue work after metadata-operations done, doe
  - directory per database, file per table to allow parallel processing of

3. FxMark Benchmark Suite
- Goal
  - Identify hot spots in file system operations
  - Measure how identified hot spots affects real applications

- Micro benchmarks
  - 18 benchmarks

- Application Benchmarks
  - filebench: fileserver, varmail, oltp workloads
  - Exim (mosbench), rocksdb (db_bench), MariaDB (tpc-c)
-UU-:@----F21  NOTE.paper     Top (39,0)    Git-master  (Fundamental Helm A
```

Fix evaluation scenarios

- Decide graphs and tables in the evaluation section
  - X-axes, y-axes
- Get baseline number first
- Then get your numbers
- Automate everything



Writing in bottom-up order

- Abstract
- Introduction
- Background and Motivation
- Design
- Implementation
- Evaluation
- Discussion and Limitation
- Related work
- Conclusion

- Evaluation
- Design
- Implementation
- Background and Motivation
- Related work
- Discussion and Limitation
- Introduction
- Conclusion
- Abstract

Writing a section

- Design the logical flow between paragraph
- Writing a paragraph
  - Conclusion first
  - Why? Provide supporting data

Evaluation Section

- ***Show you achieved all you promised***
  - ***Benefit? Design decision?***
- List up evaluation questions
  - Show your evaluation is complete
- Show benefit of your system first
  - then explain how your design decisions affects
  - X: micro-benchmark → application benchmark
  - O: application benchmark → micro-benchmark
- Draw graphs and tables

Design Section

- ***Make sure non-expert reviewers can easily understand***
- Draw a self-contained, illustrative example of your system
  - In most cases, layered diagram is useless
  - Introduce key components and their interactions of your system
  - Define terms in your system
- Explain the diagram
- Emphasize your new, cool stuff

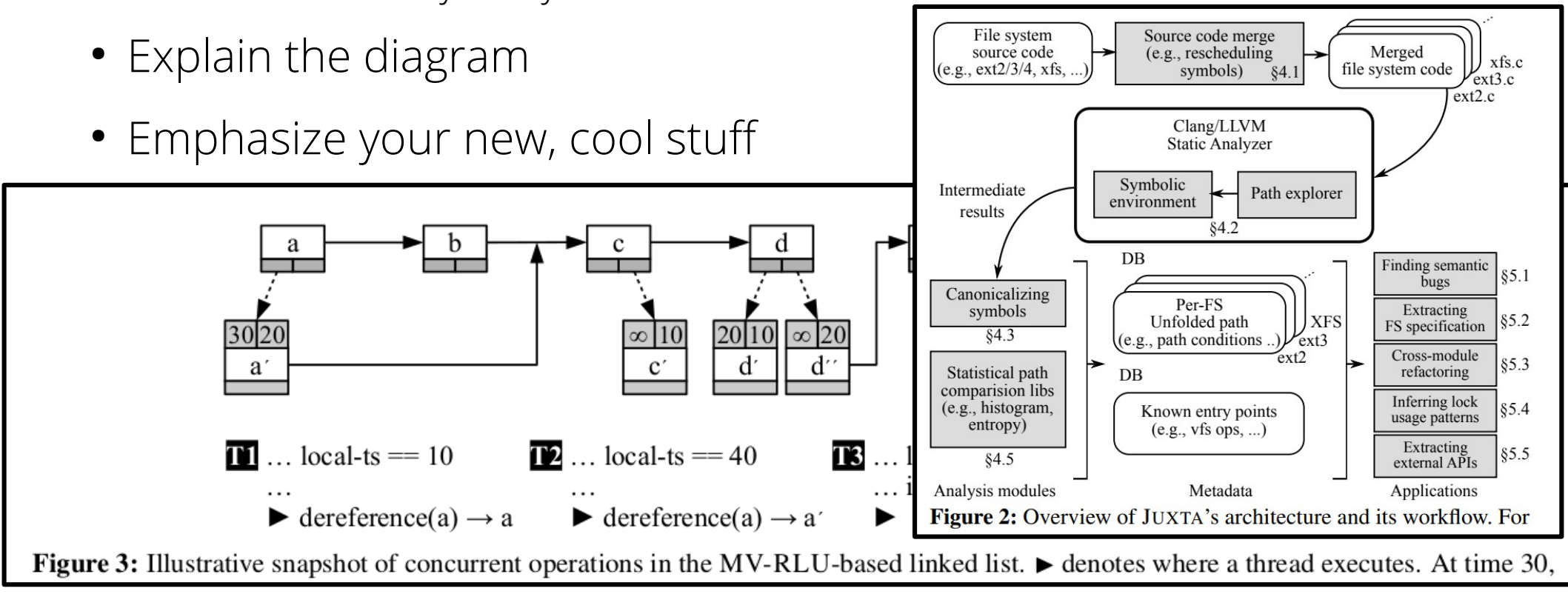


Figure 3: Illustrative snapshot of concurrent operations in the MV-RLU-based linked list. ► denotes where a thread executes. At time 30,

Figure 2: Overview of JUXTA's architecture and its workflow. For

Implementation Section

- ***Clearly separate your reusable idea and implementation artifacts***
- Design section
  - reusable idea which will be usable in 20 years later
- Implementation section
  - Implementation-specific optimization
    - Show off your implementation is solid
  - Limitation in implementation
    - != limitation in design

Background and Motivation

- **Background**
  - ***Provide enough background for non-expert reviewers to understand this paper without reading other paper***
- **Motivation**
  - ***Show fundamental limitations of existing approaches***
    - Compare performance and design decisions

Approach	Algorithm	Parallelism RR	RW	WW	Linearizable	Programming Difficulty
Lock	mutex	×	×	×	✓	medium
	rwlock	●	×	×	✓	medium
Lock-free	Harris list [29]	●	●	△	✓	high
Delegation-style	ffwd [55]	×	×	×	✓	low
	NR [11]	●	×	×	✓	low
STM	SwissTM [23]	●	▲	△	✓	lowest
	STO [35]	●	▲	△	✓	low
RCU-style	RCU [46]	●	●	×	-	high
	RLU [44]	●	●	△	-	medium
	MV-RLU	●	●	△	-	medium

1	2		synchronous log writing (`rlu_synchronize`)
1	1+1/V		version chain traversal

NOTE. ×: no parallelism ●: full parallelism ▲: read-write conflict for the same data △: write-write conflict for the same data V: number of versions at GC

Table 1: High-level comparison of synchronization mechanisms. Each mechanism has a unique design goal, strategy to scale, and target class

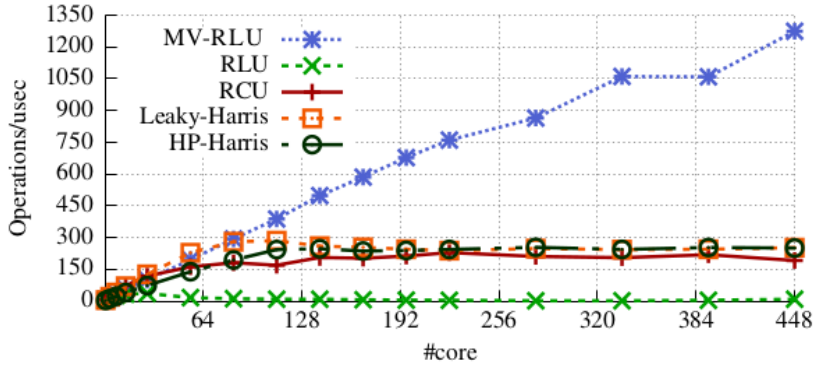


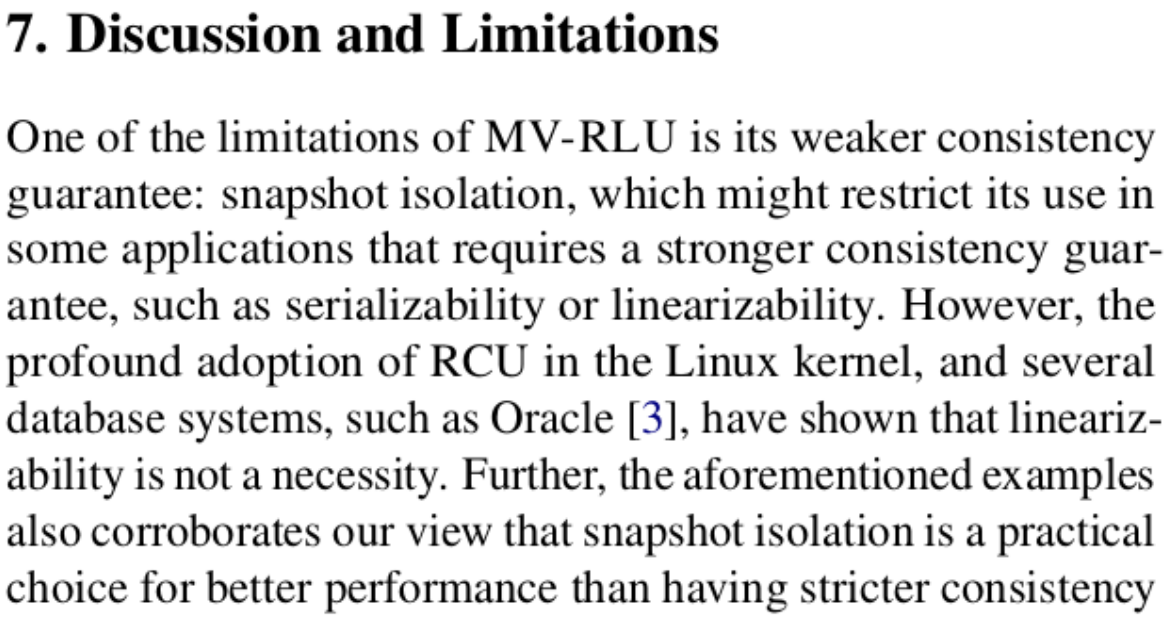
Figure 1: Performance comparison of concurrent hash tables having

Related Work

- ***Show you considered all related work***
- Categorize related papers in a few groups
  - E.g., 1) In-memory database system, 2) Software transactional memory, 3) Synchronization framework for CDS
- Don't fight against each paper
  - The authors will review your paper. ;-(
- Instead appreciate what you inherit from previous work and emphasize you enhanced
- Be precise
- Double check if there are missing related work
  - Especially, PC member's work

Discussion and Limitation

- There is no such perfect system.
- ***Show you made a informed decision even knowing your limitation.***
  - If you do not tell your limitation, reviewers will make their best efforts to figure out your limitation anyway. ;-)
- Explain why you made such decision even if you knew limitations



7. Discussion and Limitations

One of the limitations of MV-RLU is its weaker consistency guarantee: snapshot isolation, which might restrict its use in some applications that requires a stronger consistency guarantee, such as serializability or linearizability. However, the profound adoption of RCU in the Linux kernel, and several database systems, such as Oracle [3], have shown that linearizability is not a necessity. Further, the aforementioned examples also corroborates our view that snapshot isolation is a practical choice for better performance than having stricter consistency

Introduction

- How important it is
  - First 15 minutes of a movie
  - First 30 seconds of a pop song
- You SHOULD make your reviewer excited and jealous
  - “I wish I was working on this area”
  - “If I were working in that area, I wish I was working on this problem”
  - “I wish I was him/her”
- Show your enthusiasm
  - Why this is an important problem
  - Why your solution is good
  - Motivating example
- Articulate technical challenges and your contribution
  - Help review writing

Conclusion and Abstract

- Conclusion
  - Focus on what you achieved and its implication in the future
- Abstract
  - Focus on the problem
  - Introduce your cool, key idea

Revising a paper

- Ask a non-expert colleague to read
  - Ask him/her to read for an hour
  - Understand why they could not understand
  - Understand what aspects of your paper they like
- Revise the paper
- Ask an expect colleague to read
  - Ask him/her to read for 2-3 hours
  - Ask which parts are misleading
  - Understand why they could not understand
  - Understand what aspects of your paper they like
- Do not ask check grammar or spelling

Before submitting a paper

- Title! Title! Title!
  - Reviewer will choose paper only seeing the paper title
  - Misleading title → wrong reviewer assignment → bad review
- Check if every is PERFECT
  - Spelling, grammar, graphic, citations
  - Sloppiness undermines reader's trust
- Revise abstract, introduction, and conclusion

If your paper is rejected

- Paper rejection is a part of research process
  - You just got feedback from research community
- Don’t blame reviewers. It’s your fault.
  - If they didn’t understand, it was your fault for them to misunderstand.
  - If they found flaws of your system, you should thank.
  - Check if reviewers read the paper as you intended
- Finish your homework before next submission
  - Top conferences NEVER accept a paper having clear flaws.

Final thoughts

- Ask a bold question and find a simple solution
- People remember only what you finish
- Find colleagues and mentors
- Help each other
- Keep trying
- Enjoy!