Joyent

A Wardrobe for the Emperor
Stitching practical bias into systems software research

Bryan Cantrill
CTO

bryan@joyent.com

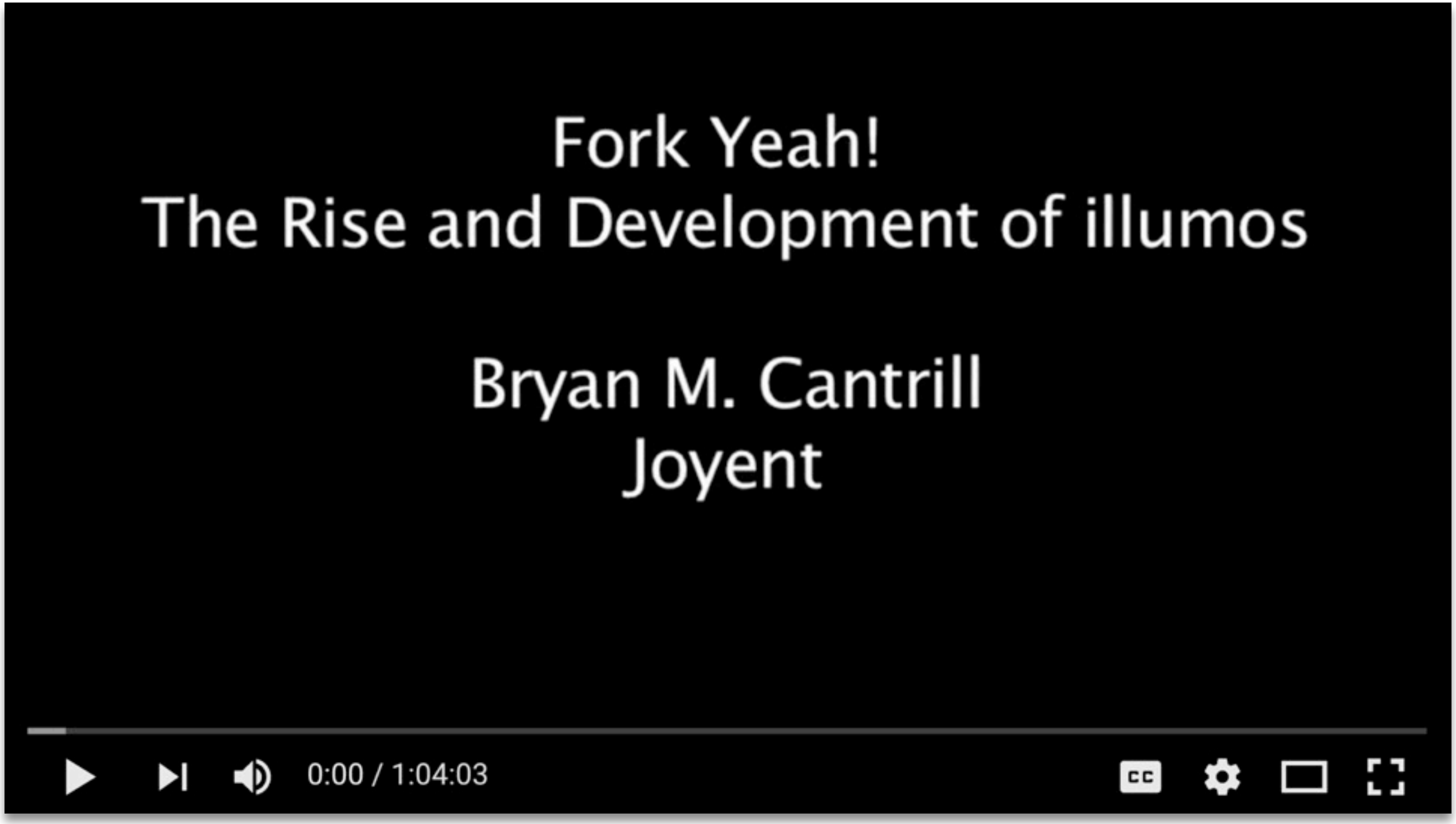
@bcantrill

Who am I?

Joyent

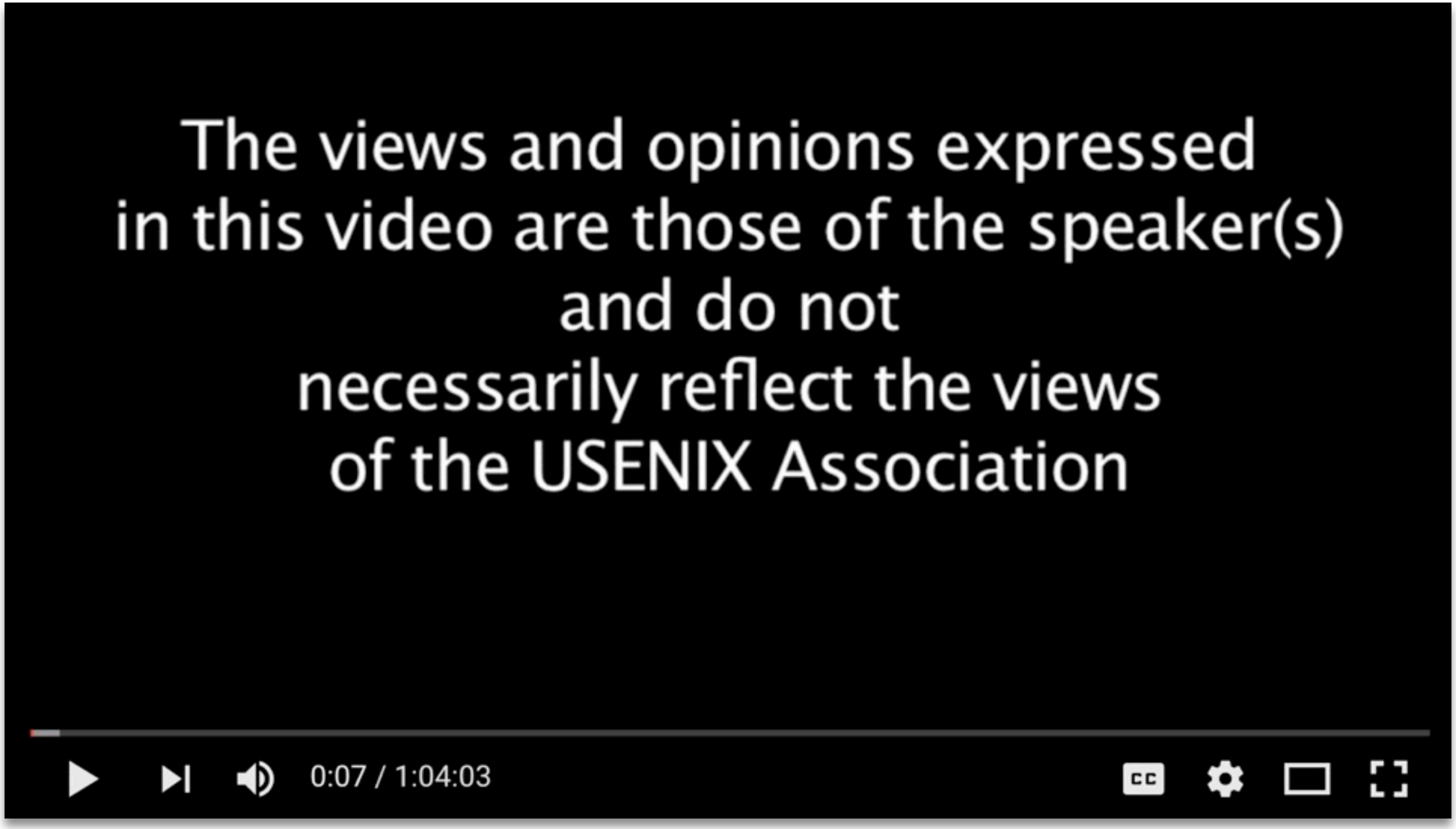
- I am a software *practitioner*: **I create production systems**
- As a practitioner on the leading edge of systems development, systems software research has always been germane...
- ...but preserving the *practical bias* in systems software research is essential to me — impractical systems aren't that helpful
- I have seen the practical bias erode over the last twenty years
- As the preeminent organization supporting practically biased systems research, USENIX can serve as a lens to understand the changes in formal systems software research...

The last time I presented at USENIX...

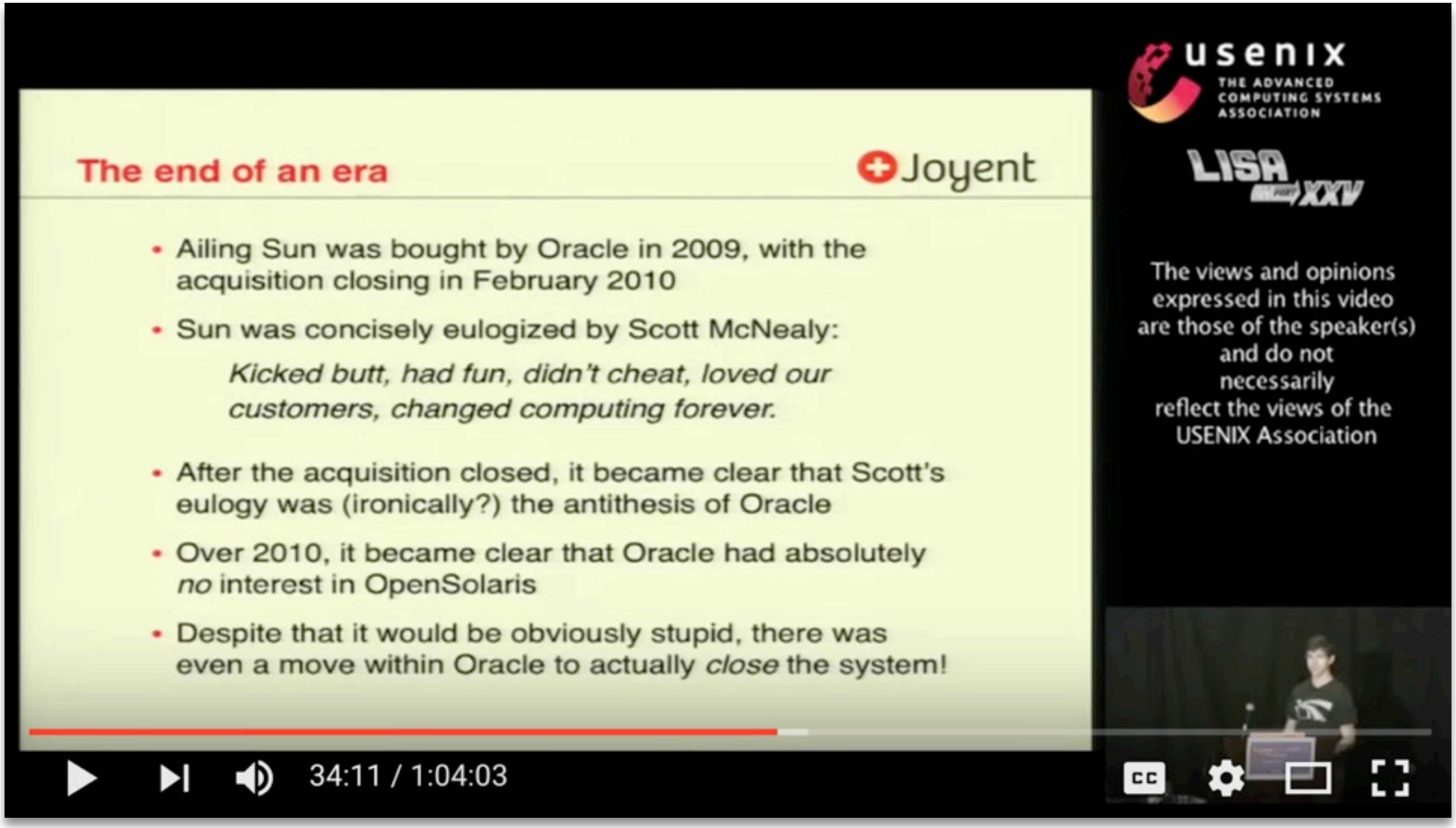

Fork Yeah!
The Rise and Development of illumos

Bryan M. Cantrill
Joyent

The last time I presented at USENIX...


The views and opinions expressed
in this video are those of the speaker(s)
and do not
necessarily reflect the views
of the USENIX Association

The last time I presented at USENIX...



The end of an era

- Ailing Sun was bought by Oracle in 2009, with the acquisition closing in February 2010
- Sun was concisely eulogized by Scott McNealy:

  Kicked butt, had fun, didn't cheat, loved our customers, changed computing forever.

- After the acquisition closed, it became clear that Scott's eulogy was (ironically?) the antithesis of Oracle
- Over 2010, it became clear that Oracle had absolutely *no* interest in OpenSolaris
- Despite that it would be obviously stupid, there was even a move within Oracle to actually *close* the system!

The views and opinions expressed in this video are those of the speaker(s) and do not necessarily reflect the views of the USENIX Association

Without further ado: NOTICE

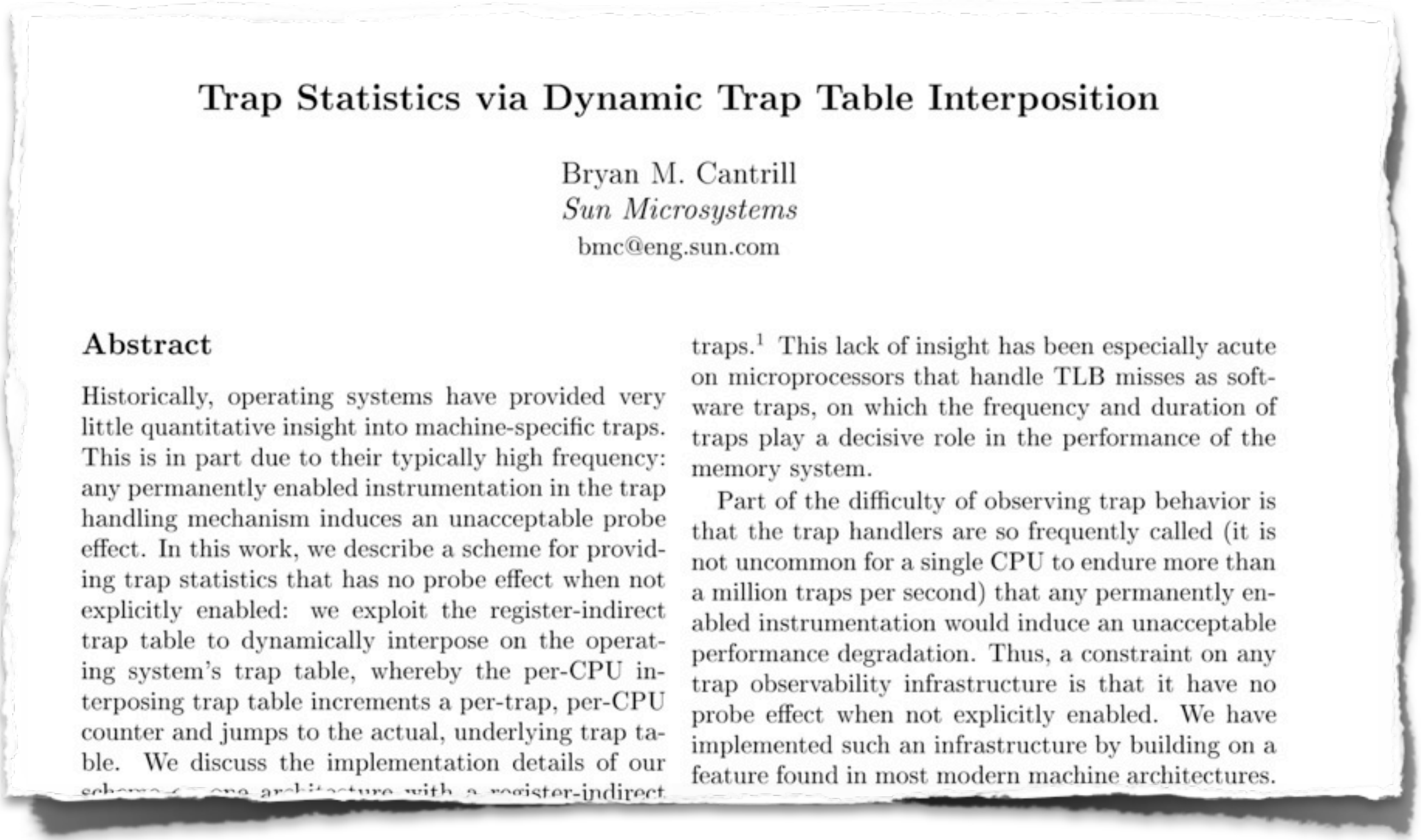
- To be clear, the views and opinions expressed in this presentation are emphatically those of the speaker, and almost certainly do **not** reflect those of the USENIX Association!
- Additionally:
  - Persons attempting to find a motive in this narrative will be prosecuted
  - Persons attempting to find a moral in it will be banished
  - Persons attempting to find a plot in it will be shot

USENIX: Back in the day

Joyent

- I came up lionizing USENIX — it's where serious practitioners published groundbreaking work
- The work described at USENIX conferences was not only rigorous, but nearly always in actual, shipping systems
- Litmus test for anyone in software systems: if you can look at the proceedings for USENIX Summer 1994 and not immediately wish you had been there, you probably shouldn't be doing this

USENIX 2003



Trap Statistics via Dynamic Trap Table Interposition

Bryan M. Cantrill
Sun Microsystems
bmc@eng.sun.com

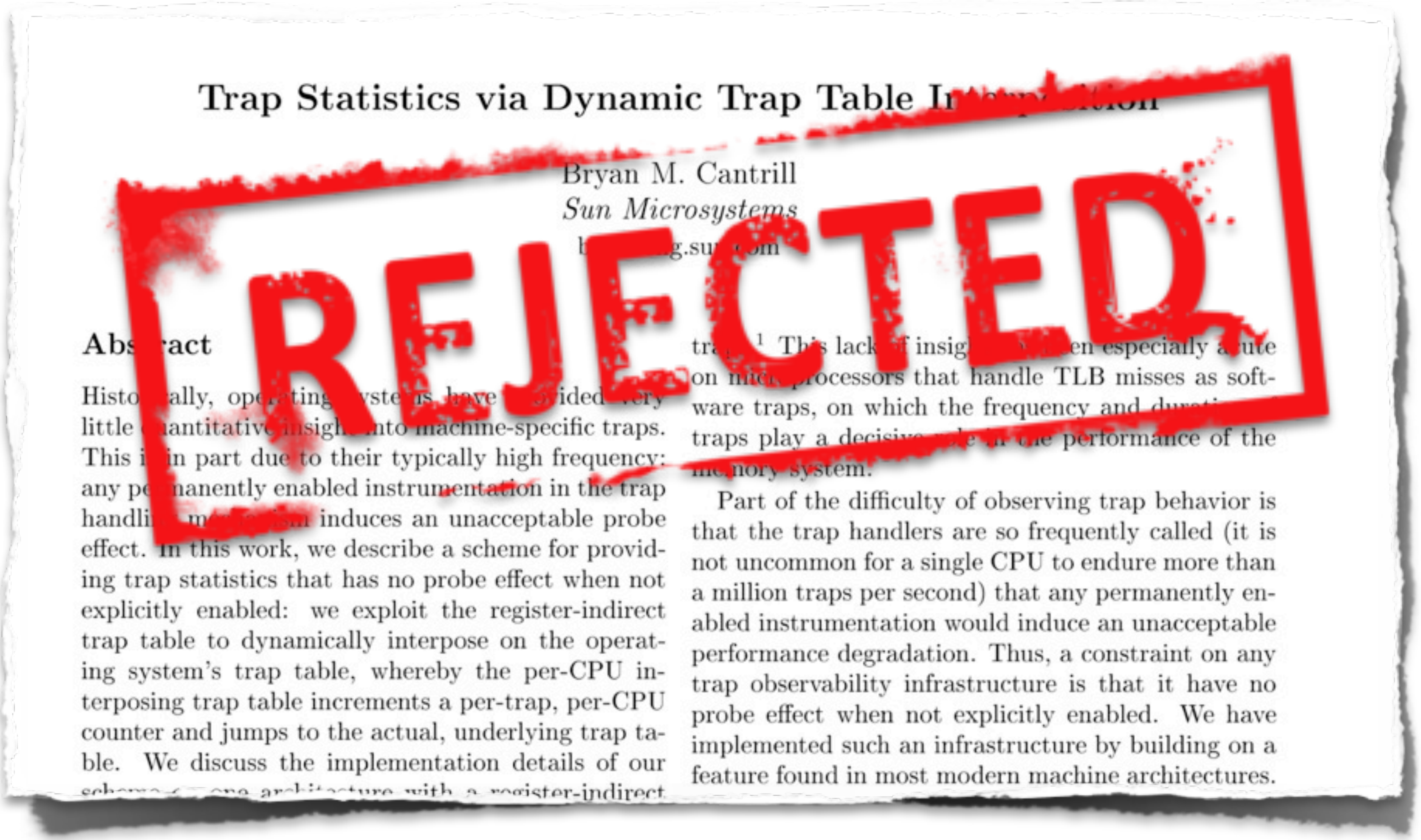
Abstract

Historically, operating systems have provided very little quantitative insight into machine-specific traps. This is in part due to their typically high frequency: any permanently enabled instrumentation in the trap handling mechanism induces an unacceptable probe effect. In this work, we describe a scheme for providing trap statistics that has no probe effect when not explicitly enabled: we exploit the register-indirect trap table to dynamically interpose on the operating system's trap table, whereby the per-CPU interposing trap table increments a per-trap, per-CPU counter and jumps to the actual, underlying trap table. We discuss the implementation details of our

traps.[1] This lack of insight has been especially acute on microprocessors that handle TLB misses as software traps, on which the frequency and duration of traps play a decisive role in the performance of the memory system.

Part of the difficulty of observing trap behavior is that the trap handlers are so frequently called (it is not uncommon for a single CPU to endure more than a million traps per second) that any permanently enabled instrumentation would induce an unacceptable performance degradation. Thus, a constraint on any trap observability infrastructure is that it have no probe effect when not explicitly enabled. We have implemented such an infrastructure by building on a feature found in most modern machine architectures.

USENIX 2003



Trap Statistics via Dynamic Trap Table Interposition

Bryan M. Cantrill
Sun Microsystems
b[illegible]g.sun.com

REJECTED

Abstract

Histo[illegible]ally, ope[illegible]ting syste[illegible]s have [illegible]vided very little [illegible]antitative insigh[illegible] into machine-specific traps. This i[illegible] in part due to their typically high frequency: any pe[illegible]manently enabled instrumentation in the trap handli[illegible] m[illegible] induces an unacceptable probe effect. In this work, we describe a scheme for providing trap statistics that has no probe effect when not explicitly enabled: we exploit the register-indirect trap table to dynamically interpose on the operating system's trap table, whereby the per-CPU interposing trap table increments a per-trap, per-CPU counter and jumps to the actual, underlying trap table. We discuss the implementation details of our sche[illegible] [illegible] one architecture with a register-indirect

tra[illegible][1] Th[illegible]s lack [illegible]f insig[illegible] [illegible]en especially a[illegible]ute on mic[illegible]processors that handle TLB misses as software traps, on which the frequency and dura[illegible] traps play a decisiv[illegible] [illegible] the performance of the me[illegible]ory system.

Part of the difficulty of observing trap behavior is that the trap handlers are so frequently called (it is not uncommon for a single CPU to endure more than a million traps per second) that any permanently enabled instrumentation would induce an unacceptable performance degradation. Thus, a constraint on any trap observability infrastructure is that it have no probe effect when not explicitly enabled. We have implemented such an infrastructure by building on a feature found in most modern machine architectures.

USENIX 2003: Reviewers’ comments

- For USENIX 2003, there were 103 submissions for 24 slots
- With the acceptance rate so low, it was no surprise to have work that was limited in scope (if novel and useful) be rejected
- But the disparate comments from the three reviewers painted a more complicated picture...

USENIX 2003: Reviewer #1

This paper describes the design and implementation of trapstat, a Solaris command that provides detailed trap count statistsics, including of TLB misses. The design has a number of interesting attributes:

-       Because the interposition is dynamic, there is no overhead if trapstat is not running.

-       The implementation makes no (or very few) assumptions on the content of the trap handlers, rather it truly interposes itself between the hardware and the standard trap handlers

-       It requires only a minor modification to the Solaris kernel itself. The rest is implemented through a loadable device driver, which includes the code that takes over the interrupt vectors.

The paper is quite detailed and thorough in its description of the mechanism. It also contains some interesting experimental results and insights; for example, the overheads of TLB miss handling in Netscape.

USENIX 2003: Reviewer #1, cont.

Suggestions for improvements: given the nice design, a port to Linux on Solaris would be interesting.   In particular any insights relating to the incremental work of the port, and any adjustments necessary for the design.

This paper is well aligned with the goals of the conference.

USENIX 2003: Reviewer #2

This paper describes a method for gather statistics on machine-specific traps by dynamically interposing data-collection code into the trap path. The author gives an example of using the method to gather statistics on TLB misses.

The paper is reasonably well written, although it has some odd English usages (see "Minor issues" below).

I have two problems with this paper. First, it seems to be too specific to the SPARC. I would be more interested if the techniques were generally applicable. Second, it seems to overlap prior work in dynamic kernel tracing. For example, Richard J. Moore's Dynamic Probes would seem to provide all of the same features and more, without being tailored to a particular architecture. There is also earlier work, although it is not as powerful as Moore's version.

USENIX 2003: Reviewer #2, cont.

```
I don't understand why the author thinks the technique is limited to
machines that have a register-indirect trap table.  Trap interposition
can be done equally well by simply replacing individual entries in a
fixed-location trap table.

The author doesn't do a good job of justifying the system.  Since he
is already modifying the kernel
```

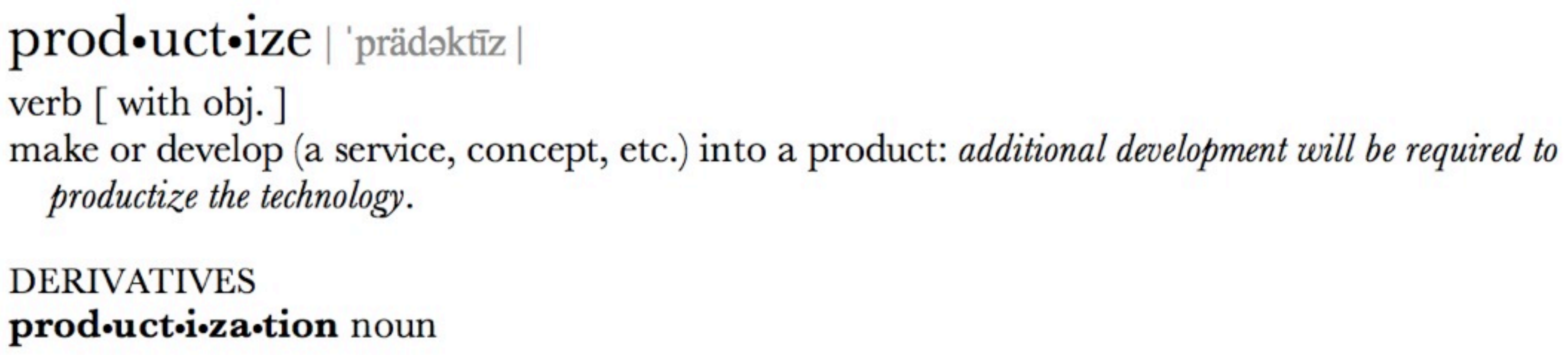
USENIX 2003: Reviewer #2, cont.

```
Minor issues:

"SPARC" and "x86" need definite or indefinite articles.  "Simpler on
x86" is incorrect usage; you should say "simpler on the x86".
```

USENIX 2003: Reviewer #2, cont.

```
"Productized" is not a word, at least not outside meetings of
marketing types who flunked English.  Try "We may turn it into a
product in Solaris x86..."
```

USENIX 2003: Reviewer #2, cont.



prod•uct•ize | ˈprädəktīz |

verb [with obj.]

make or develop (a service, concept, etc.) into a product: *additional development will be required to productize the technology.*

DERIVATIVES

prod•uct•i•za•tion noun

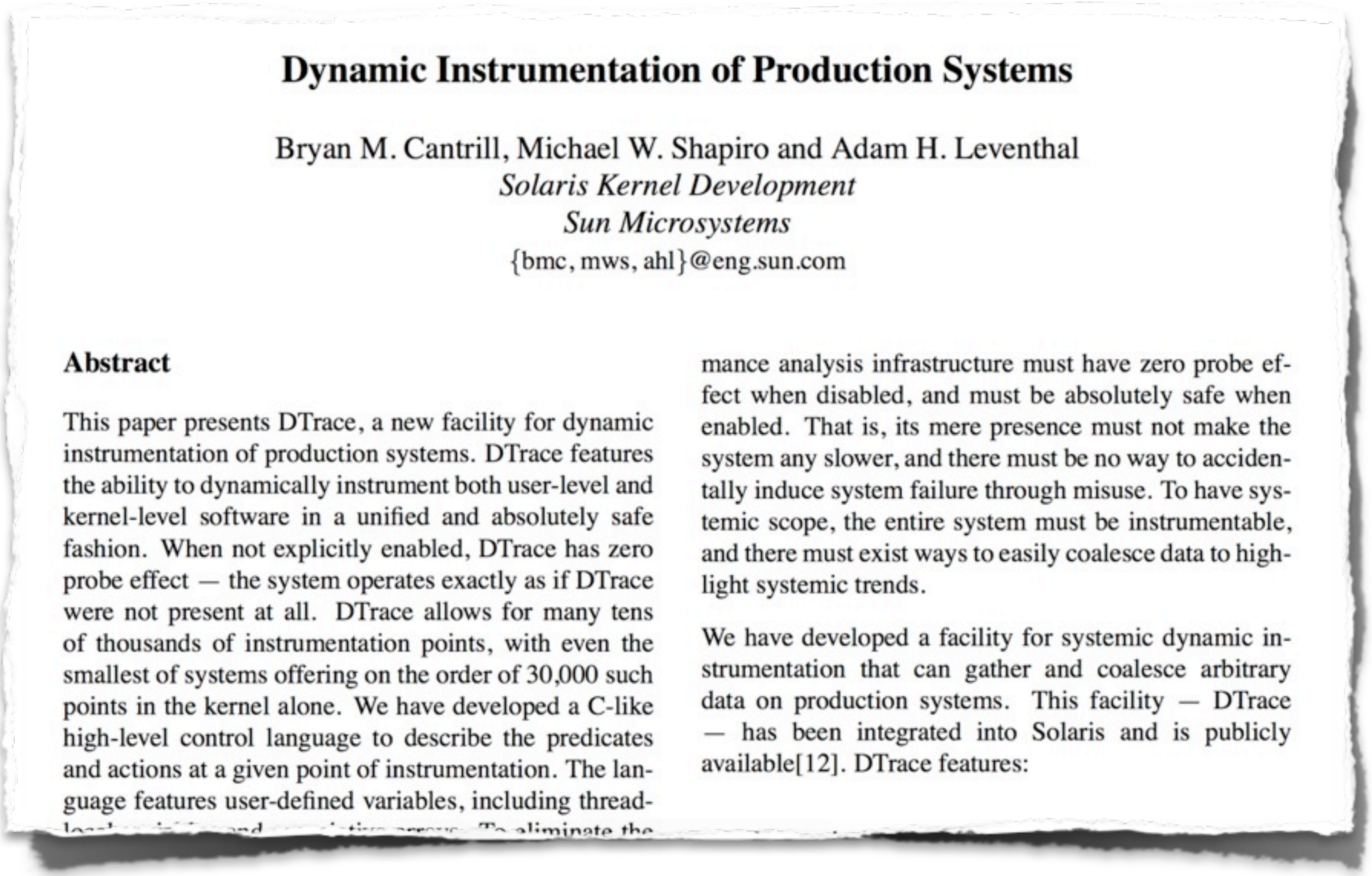
USENIX 2003: Reviewer #3

```
You should change the title to ``TLB Statistics via Dynamic Trap
Table Interposition'' since you write about naught else.

Given that you have to change the TLB handler to make this all
work, why not just have a switch in the TLB handler that means
do TLB statistics?

In section 3 you say that you use a 4 meg PTE for the trap tables
and then that the tables for each MTU live at the same virtual
address.  Does this mean that you have 4 meg of physical mem
dedicated for the trap table for each CPU?  Or am I just confused?
```

AADEBUG 2003

- USENIX 2003 experience disappointing, but not disheartening
- When the Workshop on Automated and Algorithmic Debugging (AADEBUG) announced the CFP for their 2003 conference, submitted some work on automated postmortem debugging
- Work was thoughtfully reviewed and strongly accepted by the reviewers — and the conference itself was interesting!
- AADEBUG 2003 experience inspired us to make sure that we targeted USENIX 2004 with our much more important work...

USENIX 2004



Dynamic Instrumentation of Production Systems

Bryan M. Cantrill, Michael W. Shapiro and Adam H. Leventhal
Solaris Kernel Development
Sun Microsystems
{bmc, mws, ahl}@eng.sun.com

Abstract

This paper presents DTrace, a new facility for dynamic instrumentation of production systems. DTrace features the ability to dynamically instrument both user-level and kernel-level software in a unified and absolutely safe fashion. When not explicitly enabled, DTrace has zero probe effect — the system operates exactly as if DTrace were not present at all. DTrace allows for many tens of thousands of instrumentation points, with even the smallest of systems offering on the order of 30,000 such points in the kernel alone. We have developed a C-like high-level control language to describe the predicates and actions at a given point of instrumentation. The language features user-defined variables, including thread-

mance analysis infrastructure must have zero probe effect when disabled, and must be absolutely safe when enabled. That is, its mere presence must not make the system any slower, and there must be no way to accidentally induce system failure through misuse. To have systemic scope, the entire system must be instrumentable, and there must exist ways to easily coalesce data to highlight systemic trends.

We have developed a facility for systemic dynamic instrumentation that can gather and coalesce arbitrary data on production systems. This facility — DTrace — has been integrated into Solaris and is publicly available[12]. DTrace features:

USENIX 2004

- Paper was accepted — one of only 21 out of 164 submissions!
- Even in accepting our paper, two of the reviewers were tepid; reviewer #1:

```
Overall, this is a fairly solid paper demonstrating useful extensions to
the problem domain of OS and cross-system instrumentation.  As an
application-level instrumenter it still requires further defense.
```

- Reviewer #2:

```
In terms of new contributions, it does not seem that they are many:
additions to the language (associative arrays, aggregating functions)
and speculative tracing seem like the new ones.
```

USENIX 2004

Joyent

- The third reviewer, however, was notably productive:
  - More positive (“this paper describes some nice work”)
  - Incredibly thorough (1300 words!)
  - ...and finished this way:

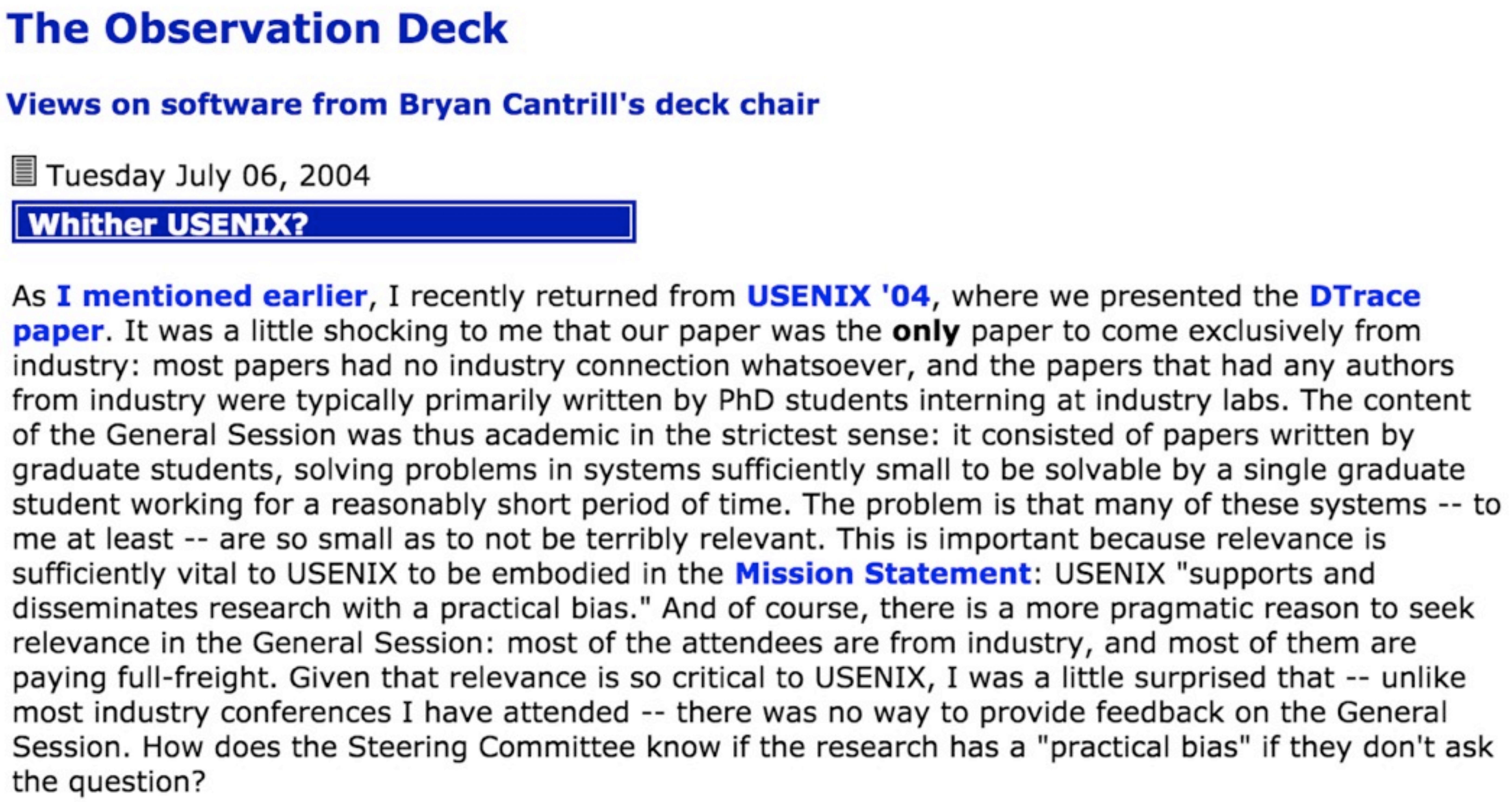
```
I hope that helps you,
Mike Burrows
```

- That he put his name to his review and wanted to help made his feedback much more meaningful!

USENIX 2004

- But the actual conference itself was disappointing: there were no other papers written by practitioners!
- The other speakers were introduced by someone saying that they were a very promising student who was looking for work (?!)
- There were few practitioners even in *attendance*; where were the 1,730 attendees from USENIX 2000?
- Where was the USENIX we knew and loved?
- The (new) blogs at Sun provided a hot mic with which to ask...

Whither USENIX?



The Observation Deck

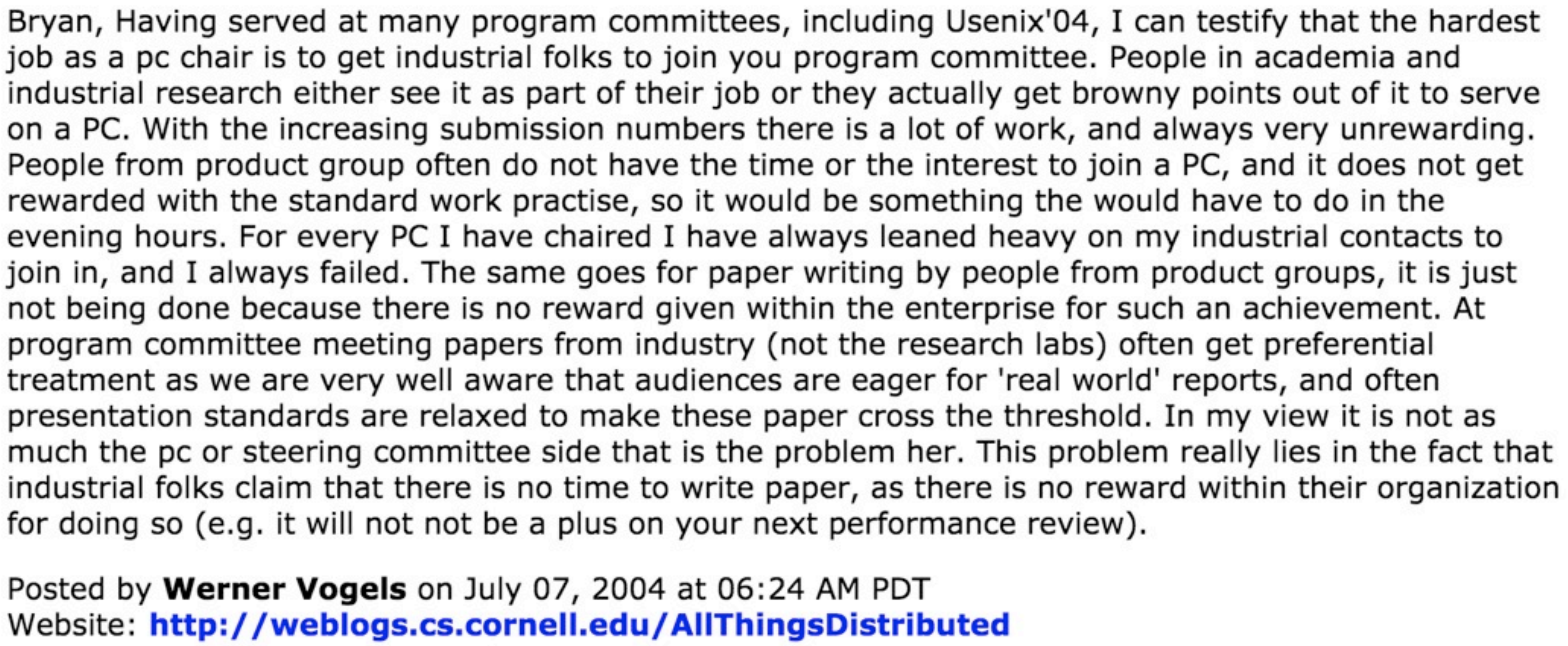
Views on software from Bryan Cantrill's deck chair

Tuesday July 06, 2004

Whither USENIX?

As **I mentioned earlier**, I recently returned from **USENIX '04**, where we presented the **DTrace paper**. It was a little shocking to me that our paper was the **only** paper to come exclusively from industry: most papers had no industry connection whatsoever, and the papers that had any authors from industry were typically primarily written by PhD students interning at industry labs. The content of the General Session was thus academic in the strictest sense: it consisted of papers written by graduate students, solving problems in systems sufficiently small to be solvable by a single graduate student working for a reasonably short period of time. The problem is that many of these systems -- to me at least -- are so small as to not be terribly relevant. This is important because relevance is sufficiently vital to USENIX to be embodied in the **Mission Statement**: USENIX "supports and disseminates research with a practical bias." And of course, there is a more pragmatic reason to seek relevance in the General Session: most of the attendees are from industry, and most of them are paying full-freight. Given that relevance is so critical to USENIX, I was a little surprised that -- unlike most industry conferences I have attended -- there was no way to provide feedback on the General Session. How does the Steering Committee know if the research has a "practical bias" if they don't ask the question?

A member of the USENIX 2004 PC responds!



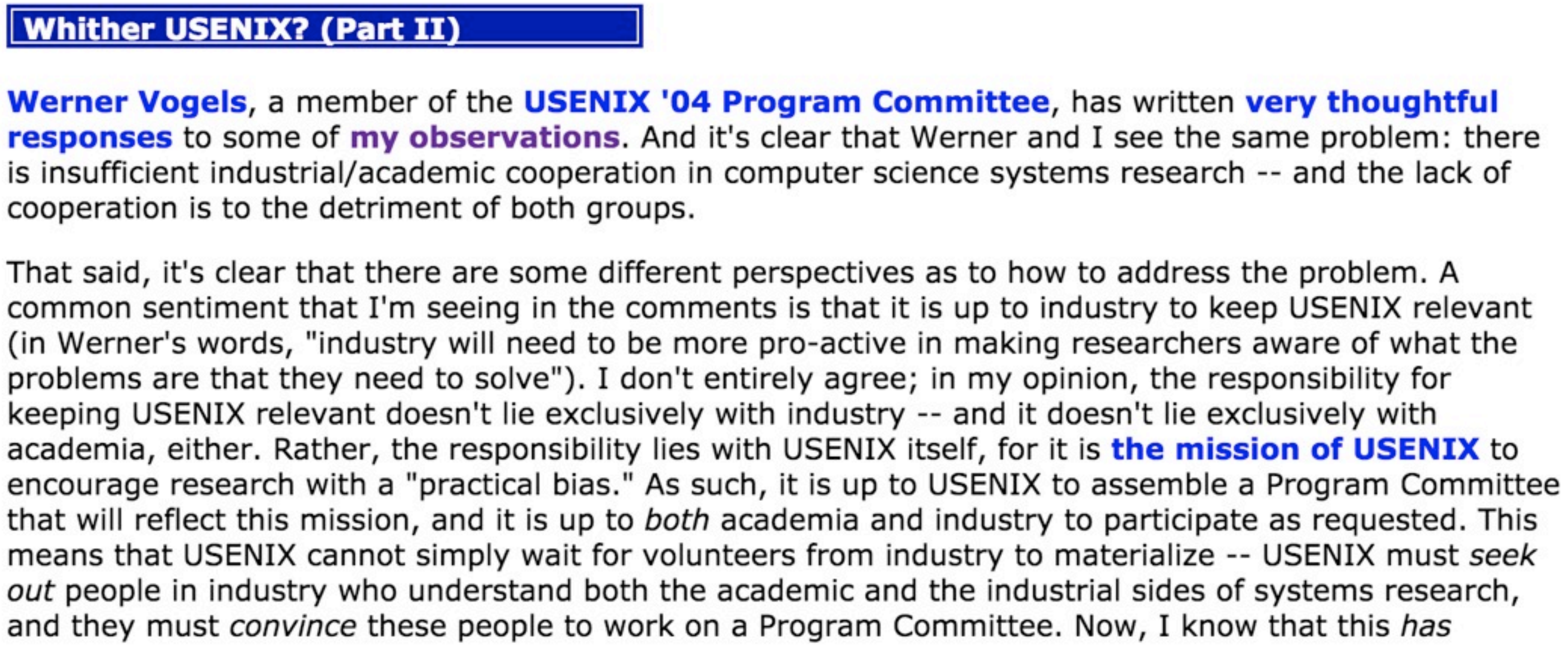
Bryan, Having served at many program committees, including Usenix'04, I can testify that the hardest job as a pc chair is to get industrial folks to join you program committee. People in academia and industrial research either see it as part of their job or they actually get browny points out of it to serve on a PC. With the increasing submission numbers there is a lot of work, and always very unrewarding. People from product group often do not have the time or the interest to join a PC, and it does not get rewarded with the standard work practise, so it would be something the would have to do in the evening hours. For every PC I have chaired I have always leaned heavy on my industrial contacts to join in, and I always failed. The same goes for paper writing by people from product groups, it is just not being done because there is no reward given within the enterprise for such an achievement. At program committee meeting papers from industry (not the research labs) often get preferential treatment as we are very well aware that audiences are eager for 'real world' reports, and often presentation standards are relaxed to make these paper cross the threshold. In my view it is not as much the pc or steering committee side that is the problem her. This problem really lies in the fact that industrial folks claim that there is no time to write paper, as there is no reward within their organization for doing so (e.g. it will not not be a plus on your next performance review).

Posted by **Werner Vogels** on July 07, 2004 at 06:24 AM PDT
Website: **http://weblogs.cs.cornell.edu/AllThingsDistributed**

Yeah, same guy

Werner Vogels, CTO of Amazon, is in charge of Amazon's cloud platform AWS. Sean Gallup/Getty Images for Burda Media

Responding to Werner

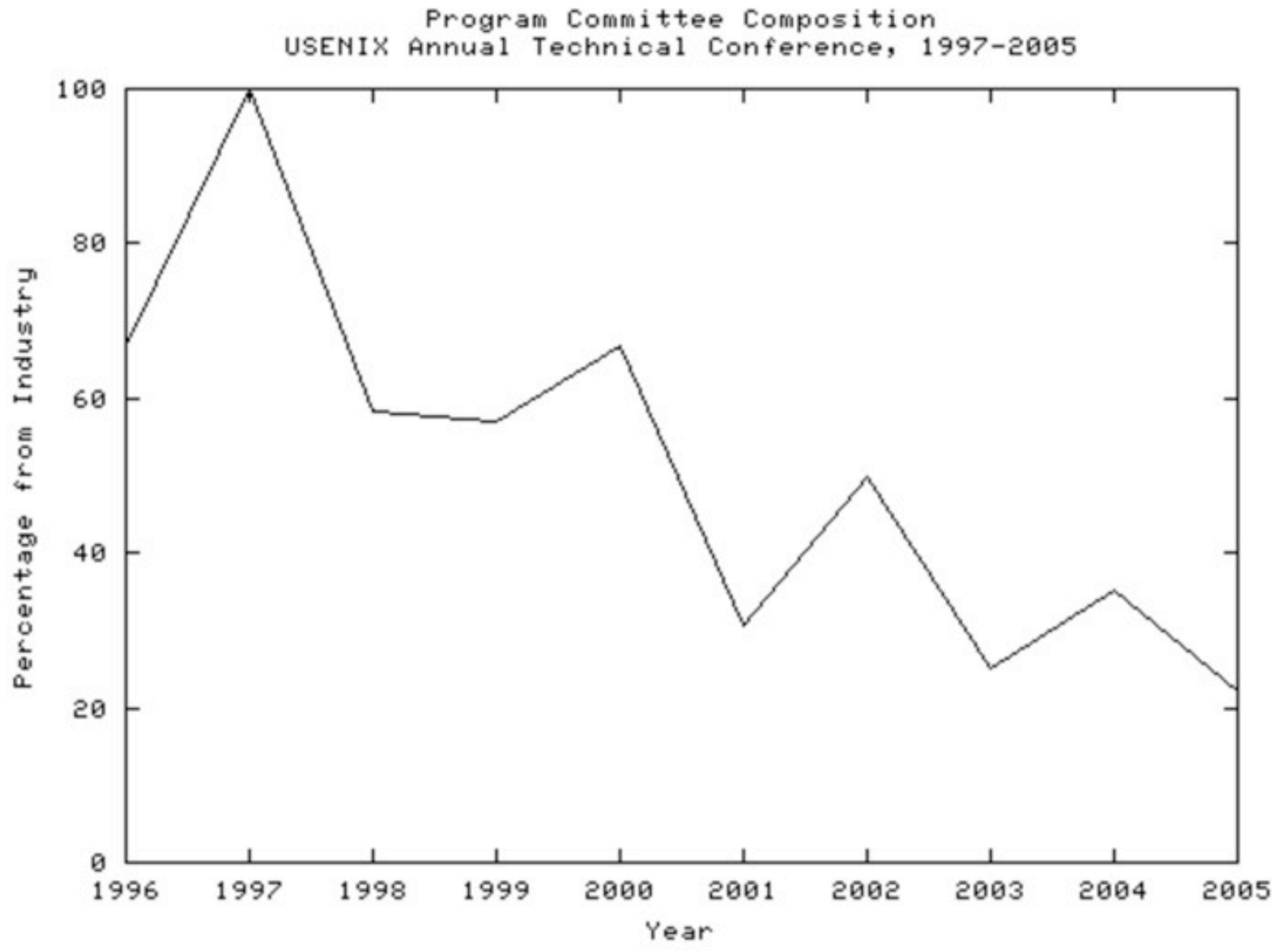


Whither USENIX? (Part II)

Werner Vogels, a member of the **USENIX '04 Program Committee**, has written **very thoughtful responses** to some of **my observations**. And it's clear that Werner and I see the same problem: there is insufficient industrial/academic cooperation in computer science systems research -- and the lack of cooperation is to the detriment of both groups.

That said, it's clear that there are some different perspectives as to how to address the problem. A common sentiment that I'm seeing in the comments is that it is up to industry to keep USENIX relevant (in Werner's words, "industry will need to be more pro-active in making researchers aware of what the problems are that they need to solve"). I don't entirely agree; in my opinion, the responsibility for keeping USENIX relevant doesn't lie exclusively with industry -- and it doesn't lie exclusively with academia, either. Rather, the responsibility lies with USENIX itself, for it is **the mission of USENIX** to encourage research with a "practical bias." As such, it is up to USENIX to assemble a Program Committee that will reflect this mission, and it is up to *both* academia and industry to participate as requested. This means that USENIX cannot simply wait for volunteers from industry to materialize -- USENIX must *seek out* people in industry who understand both the academic and the industrial sides of systems research, and they must *convince* these people to work on a Program Committee. Now, I know that this *has*

Whither USENIX: PC composition



Whither practitioners?

- Based on the (rapidly) declining involvement of practitioners in the USENIX Program Committee, it became clear that USENIX was no longer a fit for practitioners seeking to publish their work
- So if USENIX was becoming the wrong forum for practitioners to publish their work and collaborate, where could it be published?
- Fortunately, since 2004, many developments have happened that have opened up new opportunities for publishing...

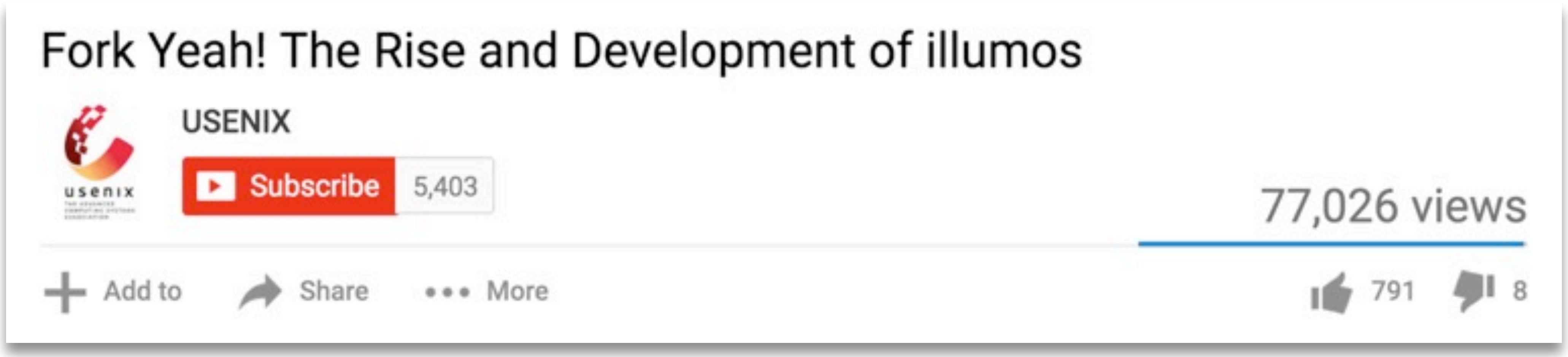
Blogging happened

Joyent

- In 2004, blogging broke into the mainstream, giving practitioners their own zero-cost publishing vehicle
- Zero-cost allows practitioners to publish small things that may be interesting to only a very small number of people
- Blogs require no fixed cadence, allowing practitioners to publish only when they have something to say
- Medium encourages candor and authenticity — a good fit for the content that practitioners want to consume

YouTube happened

- The rise of YouTube (only a decade ago!) has allowed conference content to be viewed by many more people than attend
- For most practitioners and most conferences, the conference serves as the “studio audience”: even lightly viewed videos will be viewed by more people online than in the room
- And some talks are seen by many more than could possibly ever attend a single conference:


Fork Yeah! The Rise and Development of illumos

USENIX

Subscribe 5,403

77,026 views

GitHub happened

Joyent

- Open source has been around since the dawn of computing — but the rise of GitHub has allowed for information connectedness with respect to code
- Issues can be easily filed, forks can be easily made, etc., lowering the barriers to sharing and participating in projects
- This is such a profound change that a practitioner today is unlikely to publish something meaningful without a link to a repo

ACM Queue happened

- Other leading practitioners were frustrated by the state of affairs in academic publishing: led by Steve Bourne in 2003, ACM created a new practitioner periodical, Queue
- Queue model: get leading practitioners together to brainstorm the articles they wanted to see, and then find the right practitioners to author those (peer-reviewed) articles
- No blind submissions, no program committee: a Queue author is assured that their content will be reviewed — and published
- Over the last 13 years, Queue (and CACM!) has become the home for the best practitioner-authored peer-reviewed content

Meanwhile, back in academia...

- In 2010, I was asked to join the PC for USENIX Symposium on Operating Systems Design and Implementation (OSDI) — which sits alongside SOSP as the premier systems conference
- Remembering the discussion with Werner six years prior, I felt I owed it to the discipline to give it my best effort
- By being on the inside, I hoped to answer an essential question: Could the conference model be saved for the practitioner?

OSDI ’10 Program Committee

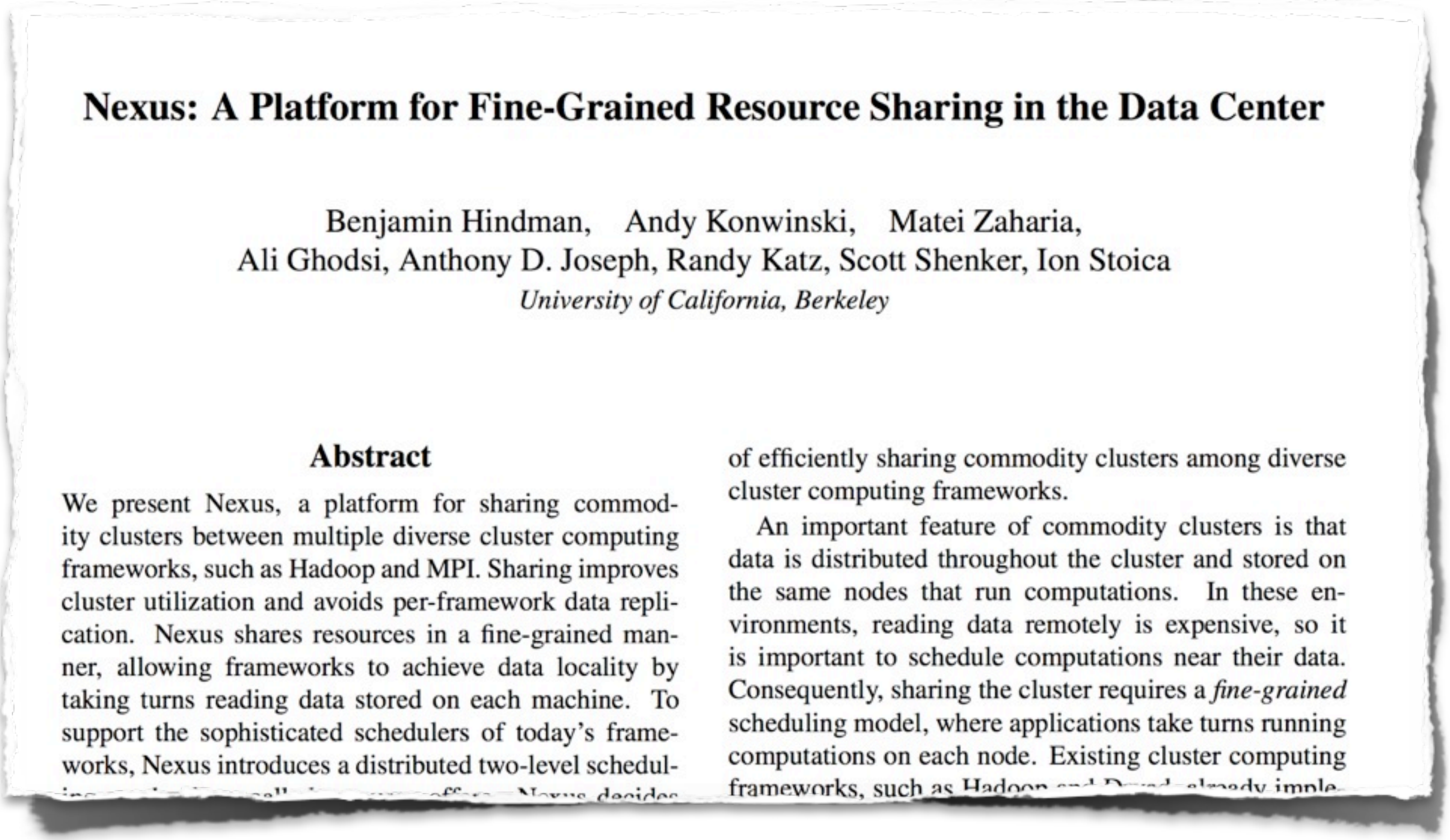
- I don’t think OSDI ’10 was atypical in its workload — and I found it to be staggering: I read (and reviewed) 36 papers!
- Of these, I wrote detailed reviews on 26 — 14,000 words in total!
- Reviewing a paper (for me, anyway) is *not* quick: 2-3 hours per paper was typical
- This was like taking 2-3 weeks off and doing nothing but reading and reviewing papers!

OSDI ’10 PC

- With some exception, the papers that I liked weren’t broadly liked: they were viewed as insufficiently novel, too small, etc.
- In general, I seemed to have greater appreciation for work that was smaller but solved a real problem — and was sufficiently polished to really test the ideas
- One of these papers that I liked that others didn’t is noteworthy...

OSDI ’10 PC

Joyent



Nexus: A Platform for Fine-Grained Resource Sharing in the Data Center

Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica
University of California, Berkeley

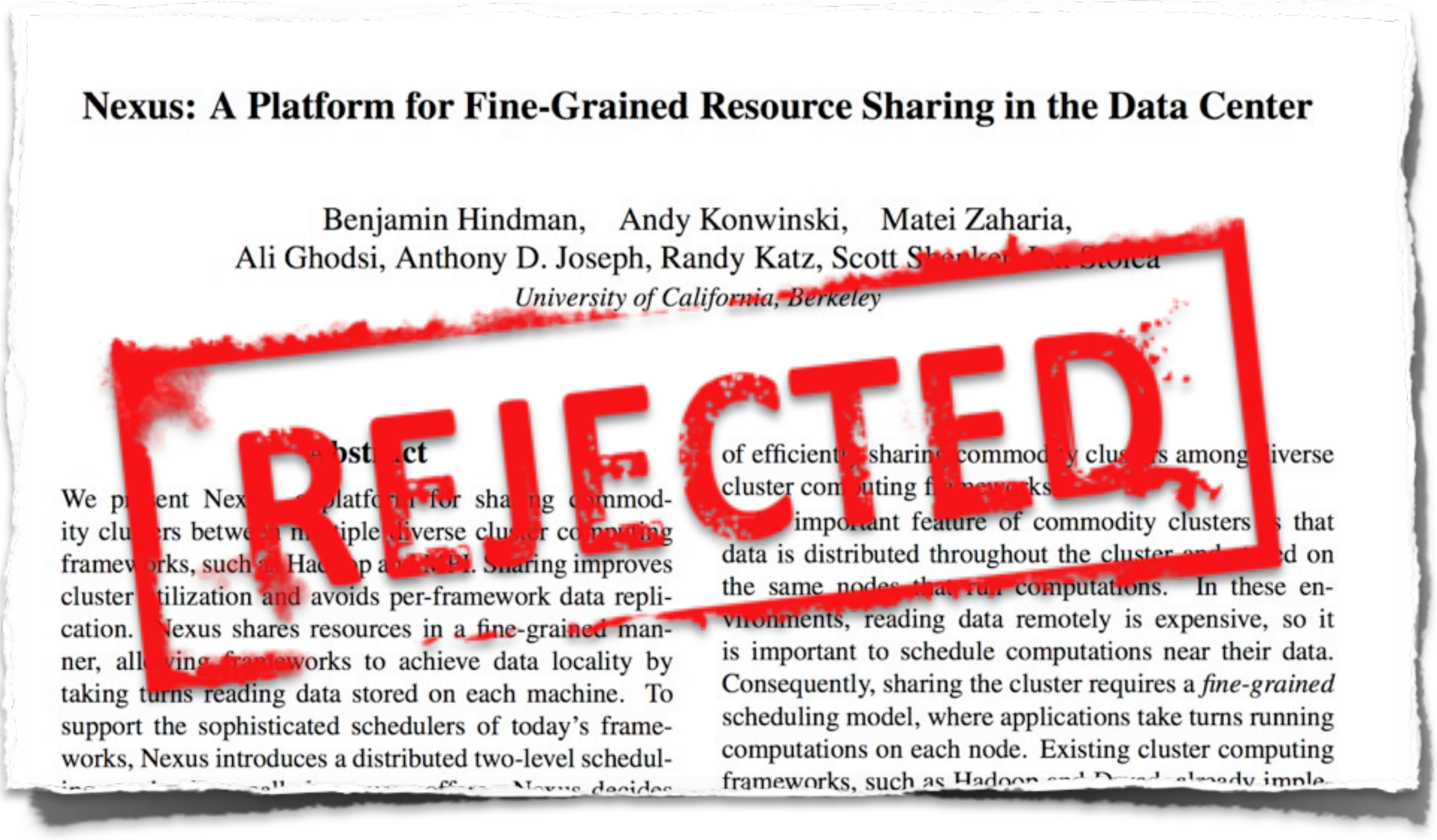
Abstract

We present Nexus, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MPI. Sharing improves cluster utilization and avoids per-framework data replication. Nexus shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To support the sophisticated schedulers of today’s frameworks, Nexus introduces a distributed two-level schedul-

of efficiently sharing commodity clusters among diverse cluster computing frameworks.

An important feature of commodity clusters is that data is distributed throughout the cluster and stored on the same nodes that run computations. In these environments, reading data remotely is expensive, so it is important to schedule computations near their data. Consequently, sharing the cluster requires a *fine-grained* scheduling model, where applications take turns running computations on each node. Existing cluster computing frameworks, such as Hadoop

OSDI ’10 PC



Nexus: A Platform for Fine-Grained Resource Sharing in the Data Center

Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica
University of California, Berkeley

REJECTED

Abstract

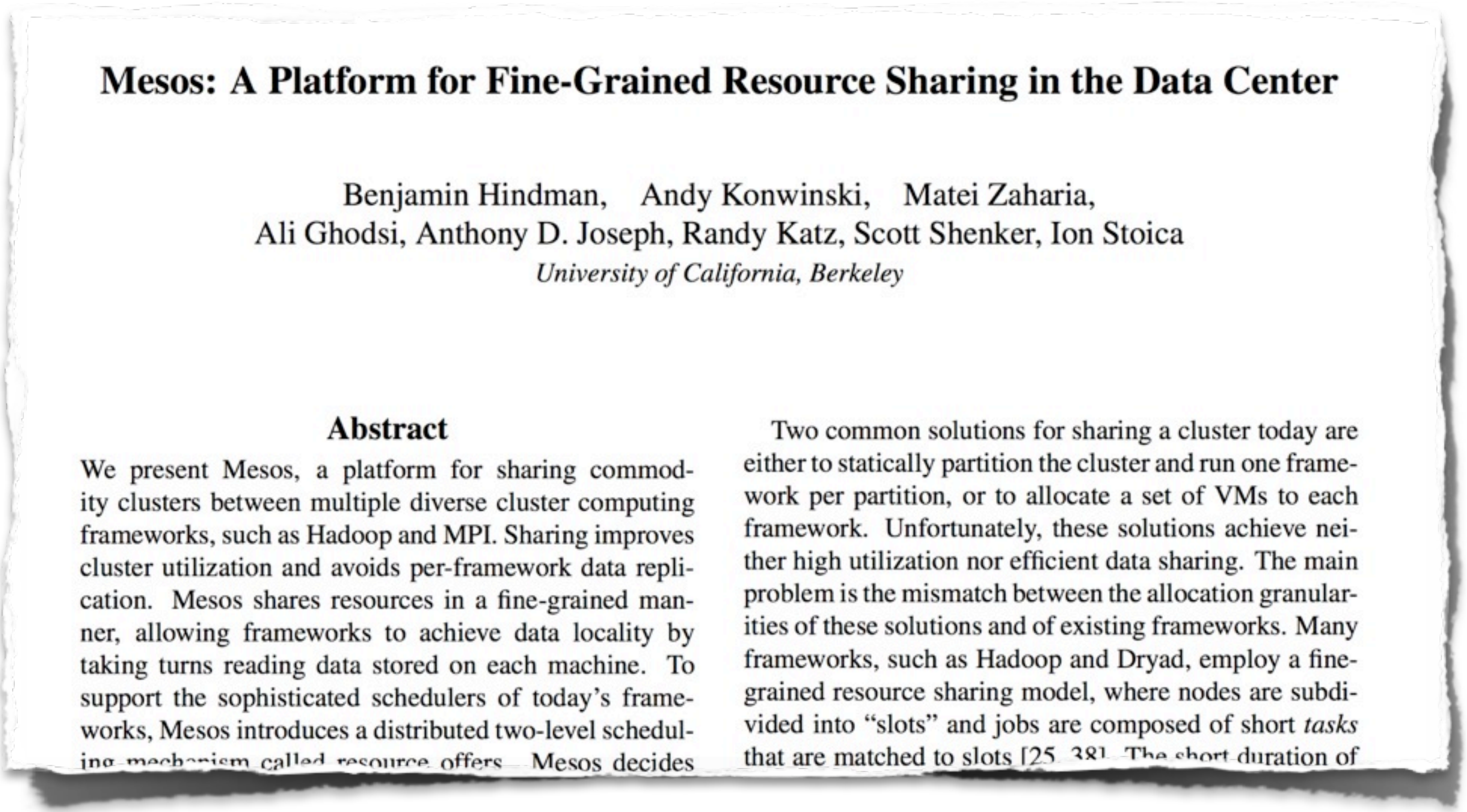
We present Nexus, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MPI. Sharing improves cluster utilization and avoids per-framework data replication. Nexus shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To support the sophisticated schedulers of today's frameworks, Nexus introduces a distributed two-level schedul-

of efficiently sharing commodity clusters among diverse cluster computing frameworks.

An important feature of commodity clusters is that data is distributed throughout the cluster and stored on the same nodes that run computations. In these environments, reading data remotely is expensive, so it is important to schedule computations near their data. Consequently, sharing the cluster requires a *fine-grained* scheduling model, where applications take turns running computations on each node. Existing cluster computing frameworks, such as Hadoop and Dryad, already imple-

You may recognize it as...

Joyent



Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center

Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

We present Mesos, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MPI. Sharing improves cluster utilization and avoids per-framework data replication. Mesos shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To support the sophisticated schedulers of today's frameworks, Mesos introduces a distributed two-level scheduling mechanism called resource offers. Mesos decides

Two common solutions for sharing a cluster today are either to statically partition the cluster and run one framework per partition, or to allocate a set of VMs to each framework. Unfortunately, these solutions achieve neither high utilization nor efficient data sharing. The main problem is the mismatch between the allocation granularities of these solutions and of existing frameworks. Many frameworks, such as Hadoop and Dryad, employ a fine-grained resource sharing model, where nodes are subdivided into "slots" and jobs are composed of short *tasks* that are matched to slots [25, 38]. The short duration of

No PC FOMO?

Joyent

- Since the Mesos paper was published (in USENIX NSDI in 2011), the work has become both popular and important
- If a VC firm had passed on the Mesos paper, they would be consumed by it: VCs have a profound fear of missing out (FOMO)
- PCs, on the other hand, do not seem to have FOMO
- Not to say that OSDI should have accepted the Nexus paper as it was, but the paper was improved by the OSDI reviewers' feedback — it's a shame we couldn't iterate and publish in OSDI!
- **Who evaluates whether a PC made the right decision?**

Back to the OSDI ’10 PC...

Joyent

- The actual meeting of the program committee happened on a particular Saturday; PC members were asked to attend in person
- Naturally, the meeting was only for those papers that merited discussion: we didn’t discuss the papers that everyone agreed should be rejected or that no one felt strongly should be accepted
- The (very few) papers that everyone agreed should be accepted also merited no real discussion...
- ...which left us with the papers for which there was dissent

OSDI ’10 PC meeting

- While others had corporate affiliations, I was one of only two practitioners in the room (~35 member PC, ~25 in the room)
- The papers that I liked had either been accepted (because everyone liked them) or rejected (because no one else did)
- I was left in a very ugly position: fighting to *reject* papers
- Of these papers that I had to fight to reject, two are noteworthy...

OSDI ’10: Paper #1

- Paper #1 tackled an important area (one in which I have expertise) that hasn’t seen much formal consideration...
- ...but it did so with some glaring, immediately disqualifying flaws
- These were undergrad-level mistakes; from my perspective, the authors either had some fundamental misunderstandings, or the writing had glaring omissions
- I was not the only one who felt this way: of the first four reviews, three of the reviewers were “strong reject”

OSDI ’10: Paper #1

- But the fourth reviewer — senior and very established but with less domain expertise in this area — was “strong accept”
- Part of the reasoning was “OSDI needs to accept more papers” and “computer science has a reputation of eating its young”
- Long, acrimonious debate in the PC meeting, with only two of us arguing strenuously to reject it (the third wasn’t in the meeting); a vote was ultimately called for...
- The program committee voted to reject it: a (silent) majority agreed with us — with the vote divided almost purely on age

OSDI ’10: Paper #2

- Paper #2 was just a terrible idea: a deeply flawed solution to a non-problem — in an area that I have a great deal of expertise
- The other reviewer (also an expert) agreed with me
- One of the PC chairs was a co-author, so the reviews were outside of the online system — and I was stunned when it came up for discussion
- The room divided again, with the same reasoning (“we need to accept more papers”). Another bitter debate. Again we voted. Again rejected.

OSDI ’10: Wrap-up

Joyent

- The PC meeting was exhausting and miserable: I hadn’t signed up for a program committee to reject papers, and I resented being thrust into the position
- Others felt I was very negative person, but I pointed out that my aggregate scores weren’t lower than anyone else’s — it’s just the stuff I liked didn’t even come up for discussion!
- Conclusion: it is very, very difficult to be a practitioner on a program committee filled with academics and researchers!

OSDI ’10: Paper #1 aftermath

- A few days after the meeting, the PC chairs mailed the PC: upon further consideration, they felt we had not accepted enough papers — and they had unilaterally accepted Paper #1 (!!)
- This had become such an obvious farce, I didn’t even care
- But other members of the PC were livid: what does the PC vote mean if the chairs can simply overrule results they don’t like?!
- If I had any last shred of doubt that being on a PC was a waste of a practitioner’s time, it was obliterated — and I couldn’t bring myself to attend OSDI ’10...

OSDI ’10: Paper #2 aftermath

- Paper #2 stayed rejected (phew, I guess?)
- About six months later, I came into a free pass to a local USENIX conference, and I sent one of the engineers on my team
- He came back enraged about a terrible paper he had seen
- As he described it I realized it was… Paper #2
- Paper #2 had been published in a subsequent conference without any real change from the OSDI submission — despite extensive feedback from us on the PC about the flaws of the scheme

OSDI ’10: Outlier or trend?

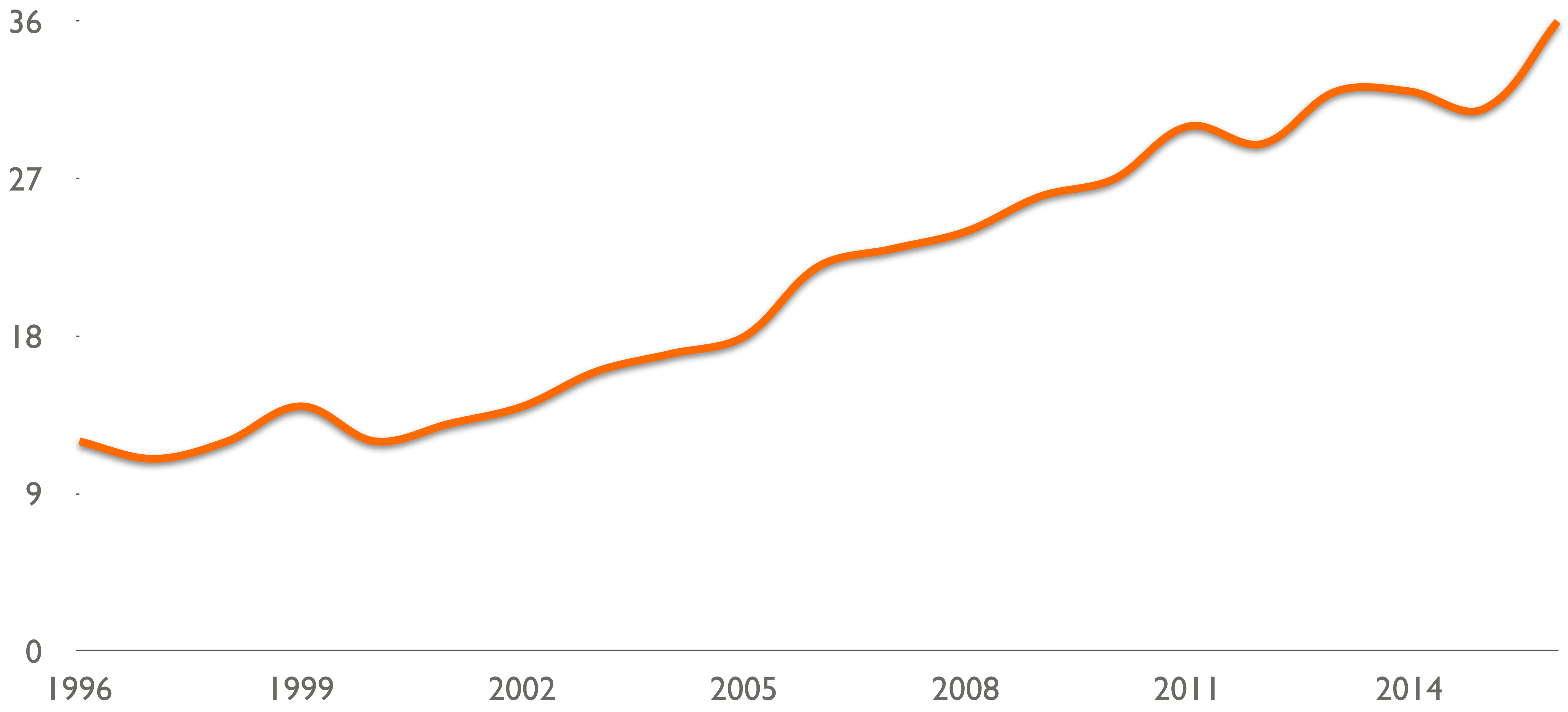
- Was OSDI ’10 “just” a bad PC? To a degree, perhaps — but several of the issues seem endemic to the model:
  - Operating under non-negotiable time pressure
  - Inability to not merely improve the writing, but get meaningful changes over an extended period of time
  - Low acceptance rates resulting in conference shopping — which further lowers the acceptance rates!
  - Low acceptance rates putting PCs under pressure to accept papers that they feel are of substandard quality

Conference model: The naked emperor

- **The conference model doesn’t work**
- It generates suboptimal research artifacts
- It deprives computer science of true conferences
- It has driven the practitioner completely away from the systems software researcher — and with it, the practical bias
- It generates unsustainable workload for program committees — who are reacting by making themselves unsustainably large!

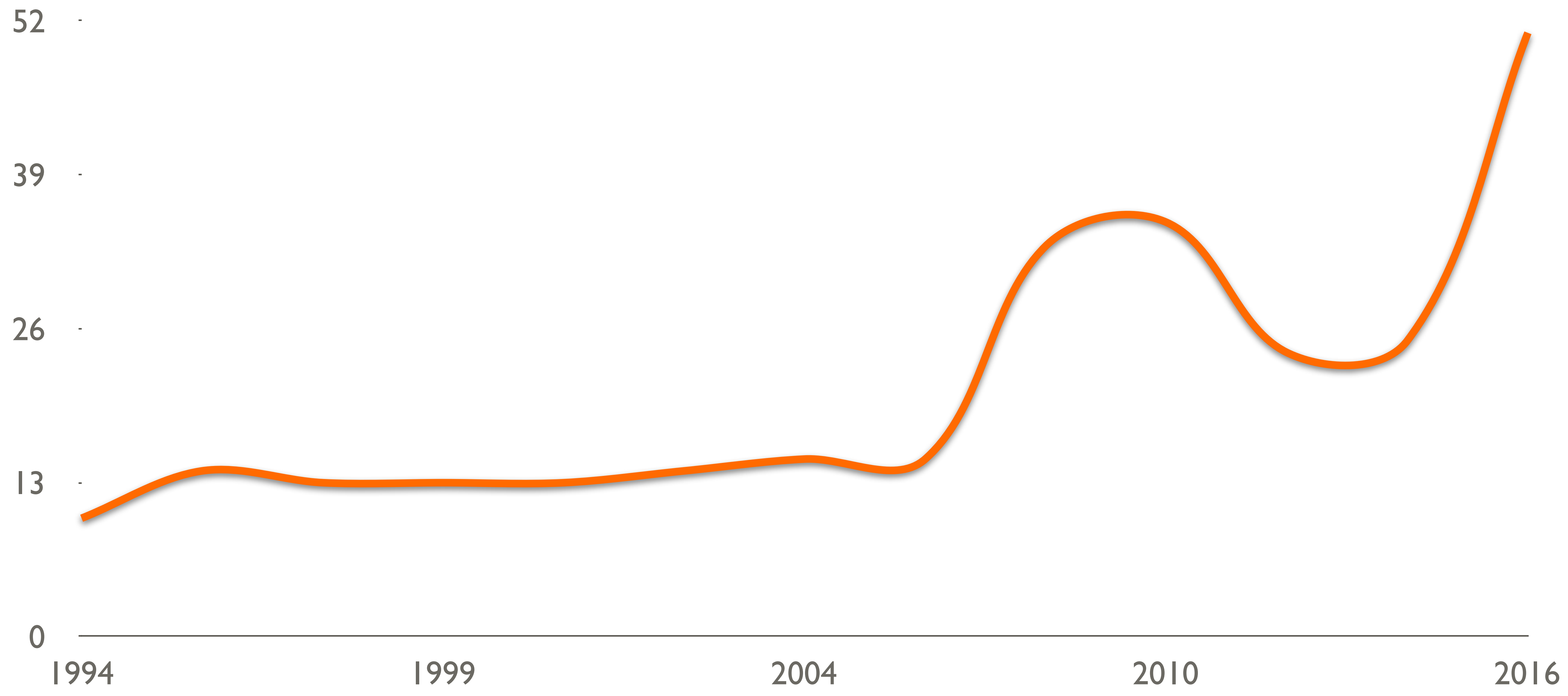
USENIX ATC: PC size over time

Joyent



OSDI ATC: PC size over time

Joyent



A new model?

- **Journals aren't the answer** — and seem likely to be disrupted by a revolution broader than just computer science
- This presents an opportunity for computer science to pioneer a *new model* that other domains could leverage
- Computer science has — by its nature — the talent within itself to solve this problem
- It seems like arXiv is a great start...

A new model?

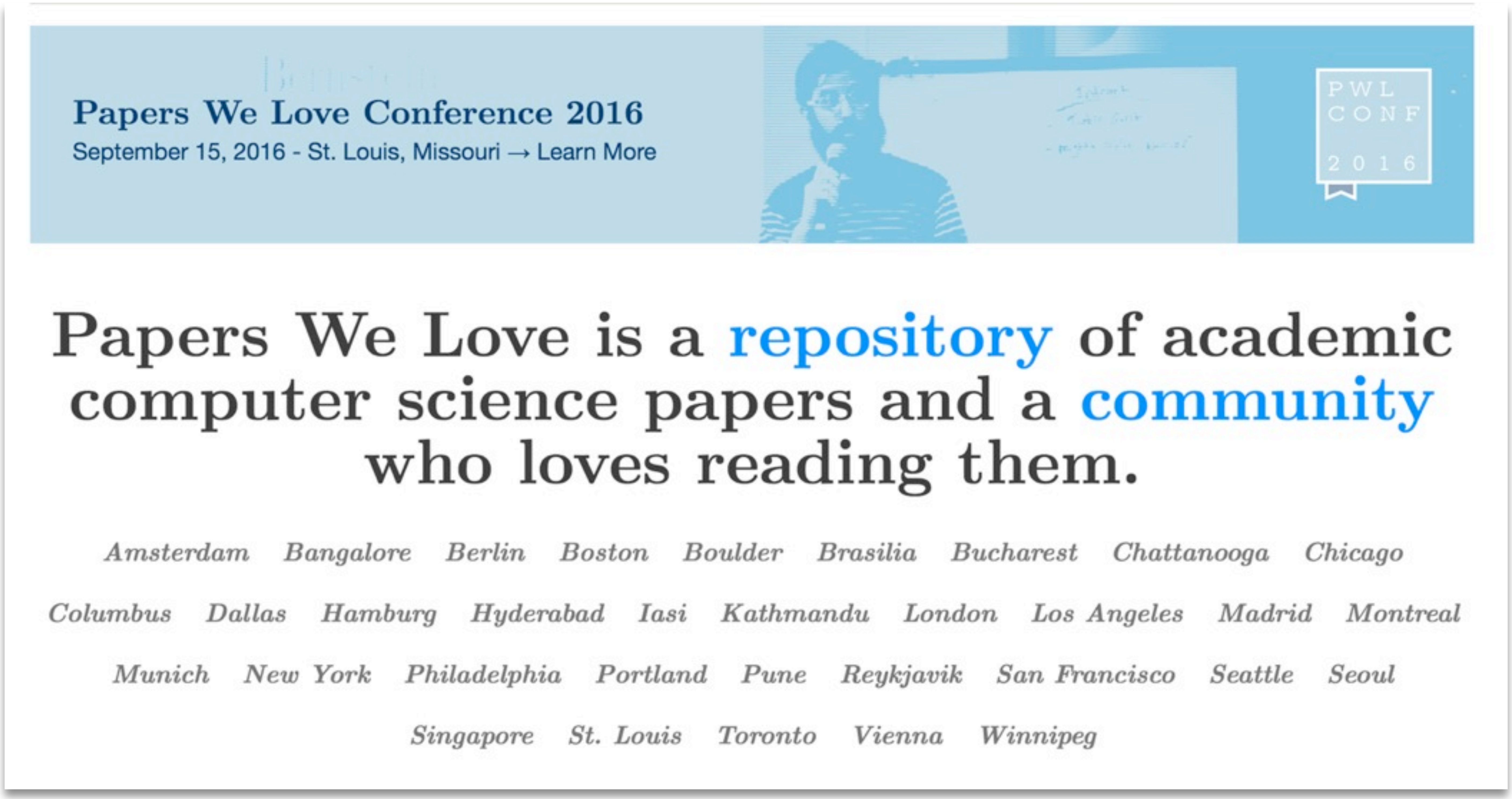
- How about a social networking aspect to arXiv? Leave reviews, get reviews, star papers that I love…
- PCs could form for the express purpose of bestowing awards on papers that they have rigorously agreed that they like
- Papers look more like films on the film festival circuit: if a paper was “accepted” by many conferences, it’s probably worth a read!
- Take a lesson from every viral social app: give badges for the behavior you want to encourage — like giving reviews on papers that the authors view as helpful!

A new model for conferences?

- Once we have solved the problem of academics and researchers being able to vet their own for purposes of hiring, promotion, grants, etc., **we can get back to actually having conferences**!
- Conferences become much more like practitioner conferences — and like conferences in other scientific domains
- Everyone goes, lots of interesting hallway conversations!
- By getting practitioners and researchers together, everyone wins: more rigorous practice, more practical research
- And yes, practitioners *are* interested in this...

Papers We Love: A reason for hope!

Joyent



Papers We Love Conference 2016

September 15, 2016 - St. Louis, Missouri → Learn More

Papers We Love is a repository of academic computer science papers and a community who loves reading them.

Amsterdam Bangalore Berlin Boston Boulder Brasilia Bucharest Chattanooga Chicago Columbus Dallas Hamburg Hyderabad Iasi Kathmandu London Los Angeles Madrid Montreal Munich New York Philadelphia Portland Pune Reykjavik San Francisco Seattle Seoul Singapore St. Louis Toronto Vienna Winnipeg

A new model for conferences

- USENIX Summer 1994 may not be coming back, but we can return to a spirit of practitioner and researcher gathering together
- For this **we need true conferences** — and we must accept that the conference model of publishing is toxic and beyond repair
- USENIX is already leading the way, but **we must be bolder**: the mandate for practical bias in its research gives USENIX the clearest case to make a revolutionary change!
- Papers We Love shows that the love for **high quality** research is very much alive — and may point the way to a new model!

Further reading

- Dan Wallach, “Rebooting the CS Publication Process”
- Bertrand Meyer, “The Nastiness Problem in Computer Science”
- Lance Fortnow, “Time for Computer Science to Grow Up”
- Batya Friedman and Fred Schneider, “Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion”
- Joseph Konstan and Jack Davidson, “Should Conferences Meet Journals and Where?”