

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Bloom: CALMly building skyscrapers on quicksand

Peter Alvaro, Neil Conway, Joseph M. Hellerstein, William R. Marczak

UC Berkeley

January 18, 2011

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- The CALM Conjecture
- Analyzing Bloom programs

3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

The future is already here

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

- Nearly all nontrivial systems are (or are becoming) distributed
- Programming distributed systems is hard
- Reasoning about them is harder

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- The CALM Conjecture
- Analyzing Bloom programs

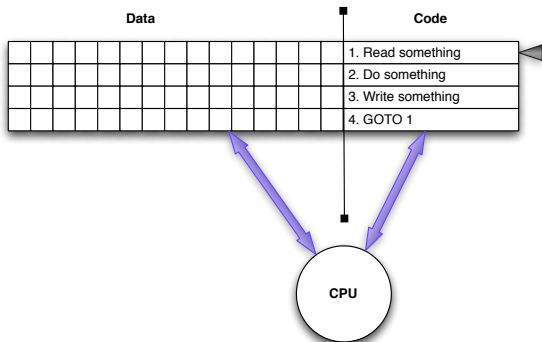
3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

The state of the art

Von Neumann model

- program state is an ordered array
- program logic is a set of ordered instructions with a PC



Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

The state of the art

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Von Neumann model – Concurrency and Parallelism

- Threads
 - copies of ordered instructions run in parallel
 - need to reason about interleavings, accesses to shared data
- Event-driven
 - single dispatching thread
 - all calls in dispatch loop must be “instant”
 - all blocking calls must be torn into top and bottom halves

The art of the state

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Disorderly Programming

- program state is unordered collections
- program logic is an unordered set of rules

The art of the state

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Disorderly Programming

- program state is unordered collections
- program logic is an unordered set of rules

Disorderly Programming – Concurrency and Parallelism

- concurrency and independence are assumed
- programmer must reason about when order is necessary
(not the opposite)

Design maxim: *think hard about the hard stuff. forget the easy stuff*

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- The CALM Conjecture
- Analyzing Bloom programs

3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

Consistency

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

A set of correctness criteria for data-oriented systems

- do my replicas all end up with the same state?
- do my partitions overlap?
- am I missing partitions?

State of the art: choose an extreme

Consistency: Choose an extreme

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

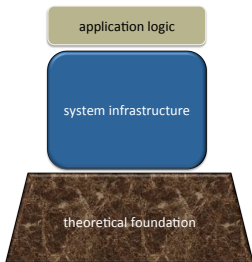
Shopping Carts

A key/value store

A destructive cart

Baked-in strong consistency via protocol (e.g. **ACID**)

- read/write serializability
- 2PC, Paxos
- **Problem:** latency, availability



Consistency: Choose an extreme

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

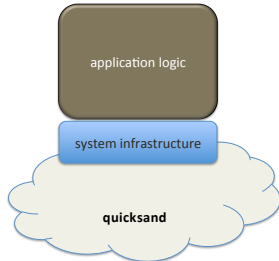
A key/value store

A destructive cart

App-specific correctness via rules of thumb (e.g. **NoSQL**)

- asynchronous replication
- commutative operations
- custom compensation / repair logic
- **Problem:** informal, fragile

"[...] all writes must be idempotent and commutative to ensure data consistency." – Gizzard Docs



Three wishes

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

- 1 A familiar, high-level language for programming distributed systems
 - encourage disorderly programming
 - but support fine-grained control of ordering when necessary
- 2 Theory and tools to reason compositionally about program correctness
 - provide guidance for when control of ordering is necessary
 - whole-program analyses: did you get your EC in my ACID?
- 3 Three more wishes...

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- The CALM Conjecture
- Analyzing Bloom programs

3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems

- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs

- 3 Shopping Carts
 - A key/value store
 - A destructive cart
 - A disorderly cart

Introducing Bloom

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

BUD: Bloom Under Development

- Ruby internal DSL
- Semantics based on Dedalus (\neg Datalog with temporal extensions)
- Set-comprehension style of programming

A BUD rule

```
multicast <~ join ([message, members]).map do |mes, mem|  
  [mem.address, mes.id, mes.payload]  
end
```

<collection>

<i>persistent</i>	table
<i>transient</i>	scratch
<i>networked transient</i>	channel
<i>scheduled transient</i>	periodic
<i>transient</i>	interface

<accumulator>

<i><=</i>	<i>now</i>
<i><+</i>	<i>next</i>
<i><-</i>	<i>del_next</i>
<i><~</i>	<i>async</i>

<collection expression>

<i><collection></i>
map, flat_map
reduce, group
join, natjoin, outerjoin
empty? include?

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- The CALM Conjecture
- Analyzing Bloom programs

3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

Abstract Interfaces and Declarations

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module DeliveryProtocol
  def state
    super
    interface input, :pipe_in,
      ['dst', 'src', 'ident'], ['payload']
    interface output, :pipe_sent,
      ['dst', 'src', 'ident'], ['payload']
  end
end
```

Concrete Implementations

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

A best-effort delivery program

```
module BestEffortDelivery
  include DeliveryProtocol

  def state
    channel : pipe_chan ,
      [ '@dst', 'src', 'ident' ], [ 'payload ' ]
  end

  declare
  def snd
    pipe_chan <~ pipe_in
  end

  declare
  def done
    pipe_sent <= pipe_in
  end
end
```

Concrete Implementations

Reliable (ack'd) delivery extends best-effort delivery, overriding the “done” declaration.

```
module ReliableDelivery
  include BestEffortDelivery

  def state
    super
    table :pipe, ['dst', 'src', 'ident'], ['payload']
    channel :ack, ['@src', 'dst', 'ident']
  end

  declare
  def remembering
    pipe <= pipe.in
  end

  declare
  def rcv
    ack <~ pipe_chan.map {|p| [p.src, p.dst, p.ident] }
  end

  declare
  def done
    apj = join [ack, pipe], [ack.ident, pipe.ident]
    pipe_sent <= apj.map {|a, p| p }
  end
end
```

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom
Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 The big idea

- Programming distributed systems
- Reasoning about distributed systems

2 Bloom

- Introducing Bloom
- Writing distributed programs in Bloom
- **The CALM Conjecture**
- Analyzing Bloom programs

3 Shopping Carts

- A key/value store
- A destructive cart
- A disorderly cart

Some insights from logic programming: terms

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Monotonic

- the more you know, the more you know. e.g.,
 - filter
 - join
 - append
- *Pipelined*

Some insights from logic programming: terms

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Nonmonotonic

- possible belief revision. Retraction. (Un-firing missiles)
 - Negation: asserting that a set does not contain a row
 - Aggregation: summarizing (e.g. counting, summing) rows
 - Deletion: removing a row from a set
- *Blocking*

Some insights from logic programming: distributed Datalog

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- We can evaluate (monotonic) Datalog in the network without any coordination
 - in spite of delay and reordering!
 - e.g., “Pipelined Semi-naive evaluation” of Loo et al
 - Monotonic \Rightarrow eventually consistent w/o coordination
- Datalog rules can be evaluated in any order and produce the same result, provided there is no negation, aggregation or deletion

CALM: Consistency and Logical Monotonicity

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 **Observation:** monotonic \Rightarrow eventually consistent
- 2 **Observation:** coordination at every nonmonotonic operation \Rightarrow EC
- 3 **Conjecture:** non-monotonic and uncoordinated \Rightarrow inconsistent

CALM Analysis

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Leverage dataflow analyses from Datalog literature to:

- Represent program as a directed graph
 - Nodes are collections
 - Arcs are rules
- Tools identify “points of order” in distributed programs
 - Arcs where different orderings of inputs may produce divergent outputs
 - Where coordination may be required to ensure consistent replicas
- Ensure that the program is well-formed
 - Free from contradictions
 - After composition, all dataflow components are connected

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems

- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs

- 3 Shopping Carts
 - A key/value store
 - A destructive cart
 - A disorderly cart

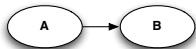
Predicate Dependency Graph Legend



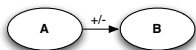
Scratch collection



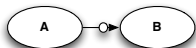
Persistent table



A appears in RHS, B in LHS of a rule R



R is a temporal rule (uses $<+$ or $<-$)



R is non-monotonic
(uses aggregation, negation, or deletion)



B is a channel



A , B , C are mutually recursive via a non-monotonic edge



Dataflow source and sink (respectively)



Indicates that dataflow is underspecified

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

Dependency Graphs

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

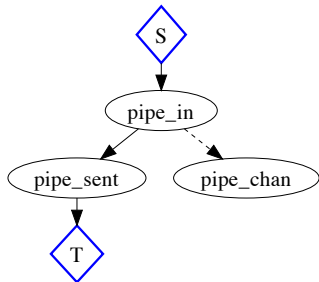
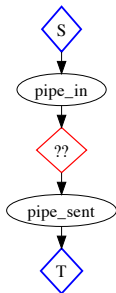
The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart



DeliveryProtocol and BestEffortDelivery

Dependency Graphs

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

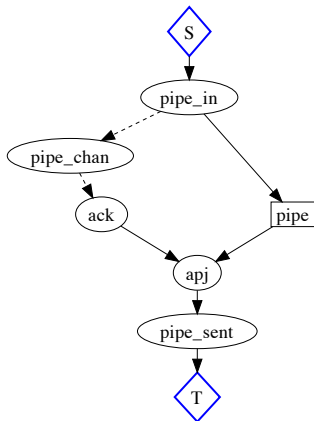
The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart



ReliableDelivery

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems
- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs
- 3 Shopping Carts
 - A key/value store
 - A destructive cart
 - A disorderly cart

Shopping Carts

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

Example application: a replicated shopping cart

- 1 Replicated to achieve high availability and low latency
- 2 Clients are associated with unique session ids
- 3 Add item, delete item and “checkout” operations

Challenge: ensure that replicas are “eventually consistent”

Rule of thumb: use commutative operations.

- Easier said than done

Carts done two ways

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

1 A “destructive” cart

- Use a replicated KVS as a storage system
- Client session id is the key
- Value is an array representing cart contents
- Checkout causes value array to be sent to client

2 A “disorderly” cart

- Accumulate (and asynchronously replicate) a set of cart actions
- At checkout, count additions and deletions, take difference
- Aggregate the summary into an array “just in time”

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems
- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs
- 3 Shopping Carts
 - A key/value store
 - A destructive cart
 - A disorderly cart

A simple key/value store

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module KVSProtocol
  def state
    super
    interface input, :kvput, ['client ', 'key ', 'reqid '], ['value ']
    interface input, :kvget, ['reqid '], ['key ']
    interface output, :kvget_response, ['reqid '], ['key ', 'value ']
  end
end
```

A simple key/value store

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module BasicKVS
  include KVSProtocol

  def state
    table :kvstate, ['key'], ['value']
  end

  declare
  def do_put
    kvstate <+ kvput.map{|p| [p.key, p.value]}
    prev = join [kvstate, kvput], [kvstate.key, kvput.key]
    kvstate <- prev.map{|s, p| s}
  end

  declare
  def do_get
    getj = join [kvget, kvstate], [kvget.key, kvstate.key]
    kvget_response <= getj.map do |g, t|
      [g.reqid, t.key, t.value]
    end
  end
end
```

CALM analysis draws:

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

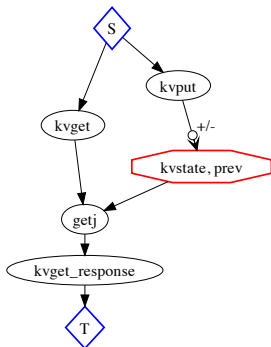
The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart



A simple key/value store

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- The analysis shows that any path through *kvput* crosses both a point of order and a temporal edge.
- Interpretation: the order of arrival of *kvput* tuples may affect the contents of *kvget_response*
- Where is the nonmonotonicity?

Where is the nonmonotonicity?

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

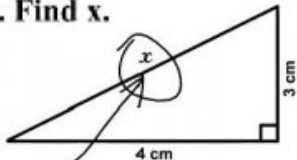
Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

3. Find x .



Here it is

A simple, conservative, syntactic check

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module BasicKVS
  include KVSProtocol

  def state
    table :kvstate, ['key'], ['value']
  end

  declare
  def do_put
    kvstate <+ kvput.map{|p| [p.key, p.value]}
    prev = join [kvstate, kvput], [kvstate.key, kvput.key]
    # dude, it's here! (<-)
    kvstate <- prev.map{|b, p| b}
  end

  declare
  def do_get
    getj = join [kvget, kvstate], [kvget.key, kvstate.key]
    kvget_response <= getj.map do |g, t|
      [g.reqid, t.key, t.value]
    end
  end
end
```

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems
- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs
- 3 Shopping Carts
 - A key/value store
 - **A destructive cart**
 - A disorderly cart

Destructive Cart

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module DestructiveCart
  include CartProtocol
  include KVSProtocol

  declare
  def do_action
    kvget <= action_msg.map{|a| [a.reqid, a.key]}
    kvput <= action_msg.map do |a|
      if a.action == "Add"
        unless kvget_response.map{|b| b.key}.include? a.session
          [a.server, a.client, a.session, a.reqid, [a.item]]
        end
      end
    end
    old_state = join [kvget_response, action_msg],
      [kvget_response.key, action_msg.session]
    kvput <= old_state.map do |b, a|
      if a.action == "Add"
        [a.server, a.client, a.session, a.reqid, b.value.push(a.item)]
      elsif a.action == "Del"
        [a.server, a.client, a.session, a.reqid, delete_one(b.value, a.item)]
      end
    end
  end
end

[...]
```

Destructive Cart – Part 2

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
[...]
```

```
declare
def do_checkout
  kvget <= checkout_msg.map{|c| [c.reqid, c.session]}
  lookup = join [kvget_response, checkout_msg],
    [kvget_response.key, checkout_msg.session]
  response_msg <~ lookup.map do |r, c|
    [c.client, c.server, c.session, r.value]
  end
end
end
```

```
end
```

Destructive Cart

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

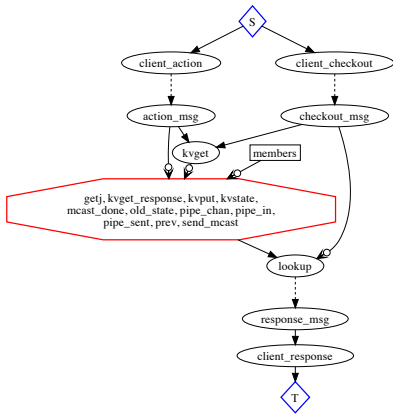
The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart



Destructive cart analysis

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

- 1 There is a point of order at each cart update from the client
- 2 There is a point of order as each tuple is forwarded to replicas
- 3 All this was evident even in the abstract KVS

Destructive cart analysis

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

Solutions:

- 1 Assert that all operations commute, and leave as is.
 - Informal and bug-prone
 - E.g., a deletion for a given item doesn't commute with that item's addition.
- 2 Add a round of distributed coordination for each update
 - E.g., Two-phase commit, Paxos
 - Overkill?
- 3 Is there a better cart abstraction?

Outline

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

- 1 The big idea
 - Programming distributed systems
 - Reasoning about distributed systems

- 2 Bloom
 - Introducing Bloom
 - Writing distributed programs in Bloom
 - The CALM Conjecture
 - Analyzing Bloom programs

- 3 Shopping Carts
 - A key/value store
 - A destructive cart
 - A disorderly cart

A simple skeleton for a “disorderly” cart

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

```
module DisorderlyCart
  include CartProtocol

  def state
    table :cart_action, ['session', 'item', 'action', 'reqid']
    table :action_cnt, ['session', 'item', 'action'], ['cnt']
    scratch :status, ['server', 'client', 'session', 'item'], ['cnt']
  end

  declare
  def do_action
    cart_action <= action_msg.map do |c|
      [c.session, c.item, c.action, c.reqid]
    end
    action_cnt <= cart_action.group(
      [cart_action.session, cart_action.item, cart_action.action],
      count(cart_action.reqid))
  end

  [...]
end
```

A simple skeleton for a “disorderly” cart – Part 2

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

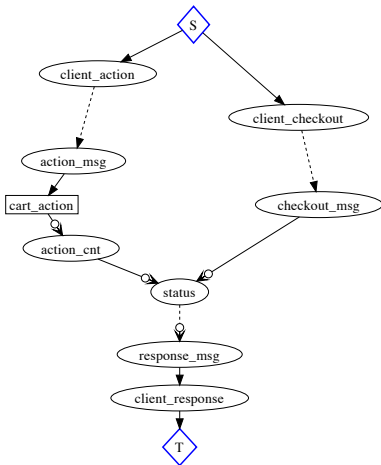
```
[...]

declare
def do_checkout
  del_items = action_cnt.map{|a| a.item if a.action == "Del"}
  status <= join([action_cnt, checkout_msg]).map do |a, c|
    if a.action == "Add" and not del_items.include? a.item
      [c.client, c.server, a.session, a.item, a.cnt]
    end
  end

  status <= join([action_cnt, action_cnt,
                checkout_msg]).map do |a1, a2, c|
    if a1.session == a2.session and a1.item == a2.item and
       a1.session == c.session and
       a1.action == "Add" and a2.action == "Del"
      [c.client, c.server, c.session, a1.item, a1.cnt - a2.cnt]
    end
  end

  response_msg <~ status.group(
    [status.client, status.server, status.session],
    accum(status.cnt.times.map{status.item}))
end
end
```

Disorderly skeleton analysis



Note the points of order (circles) corresponding to aggregation in the dataflow.

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

Replicated disorderly cart

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

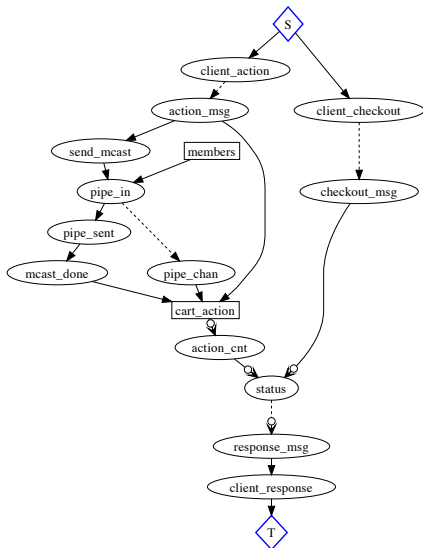
The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart



Disorderly cart analysis

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart

- 1 Our implementation is fully specified
- 2 Our concrete implementation has the same points of order that were implied by the abstraction
- 3 Client updates and replication of cart state may be coordination-free
- 4 Some coordination may be necessary to handle a *checkout* message

Review: two abstract carts

Bloom:
CALMly
building
skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

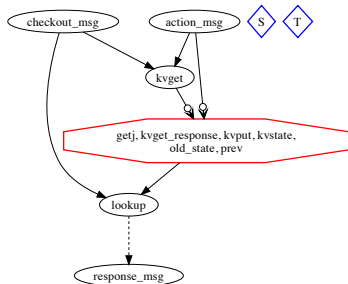
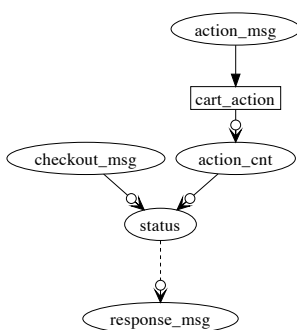
The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart



Future Directions

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping
Carts

A key/value store

A destructive cart

- Tools to guide a programmer from the 1st to 2nd implementation
- Disorderly debugging: whence?

Thanks!

Bloom:
CALMly
building

skyscrapers on
quicksand

Peter Alvaro,
Neil Conway,
Joseph M.
Hellerstein,
William R.
Marczak

The big idea

Programming
distributed systems

Reasoning about
distributed systems

Bloom

Introducing Bloom

Writing distributed
programs in Bloom

The CALM
Conjecture

Analyzing Bloom
programs

Shopping Carts

A key/value store

A destructive cart