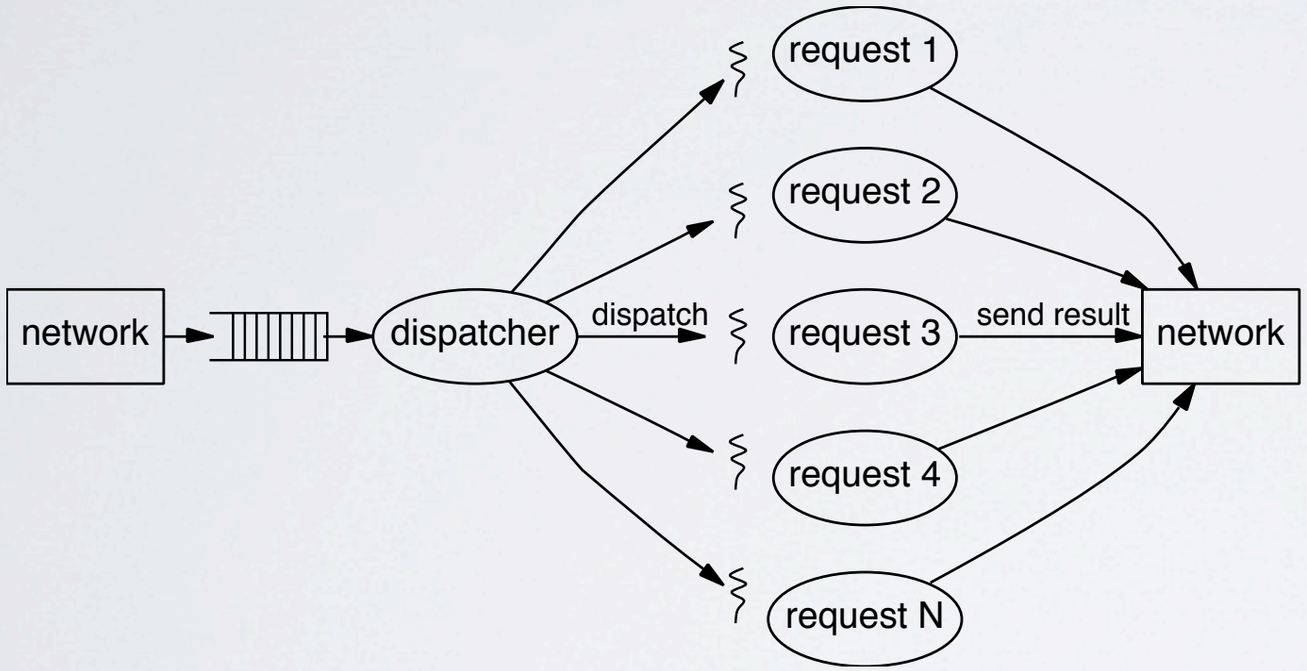
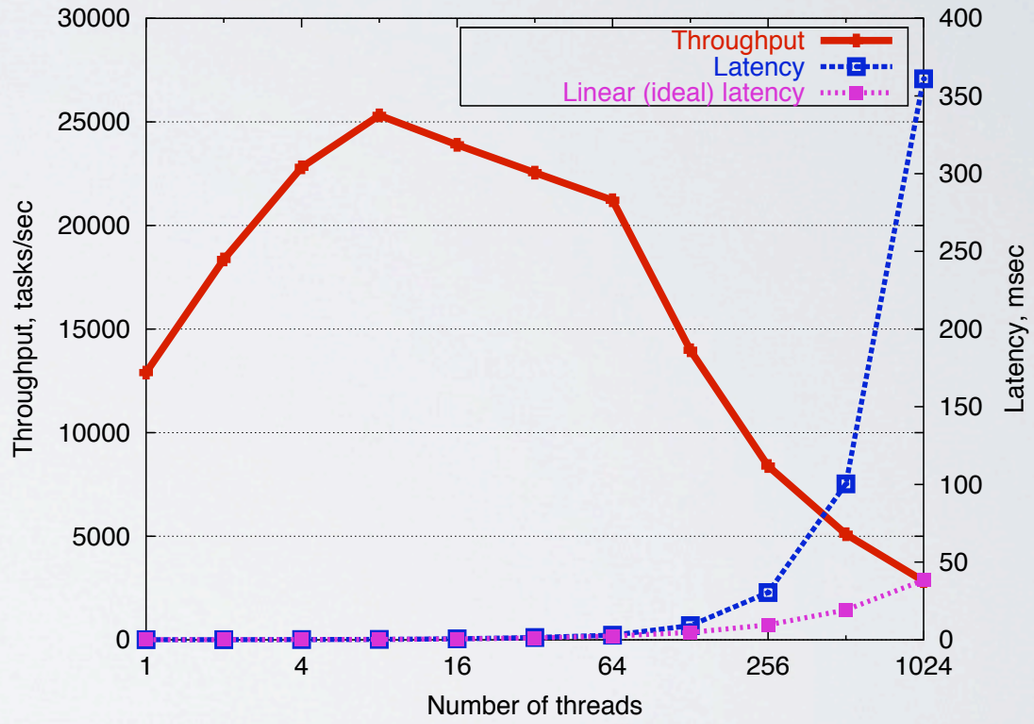
WHY EVENTS ARE A BAD IDEA

Rob von Behren, Jeremy Condit, Eric Brewer

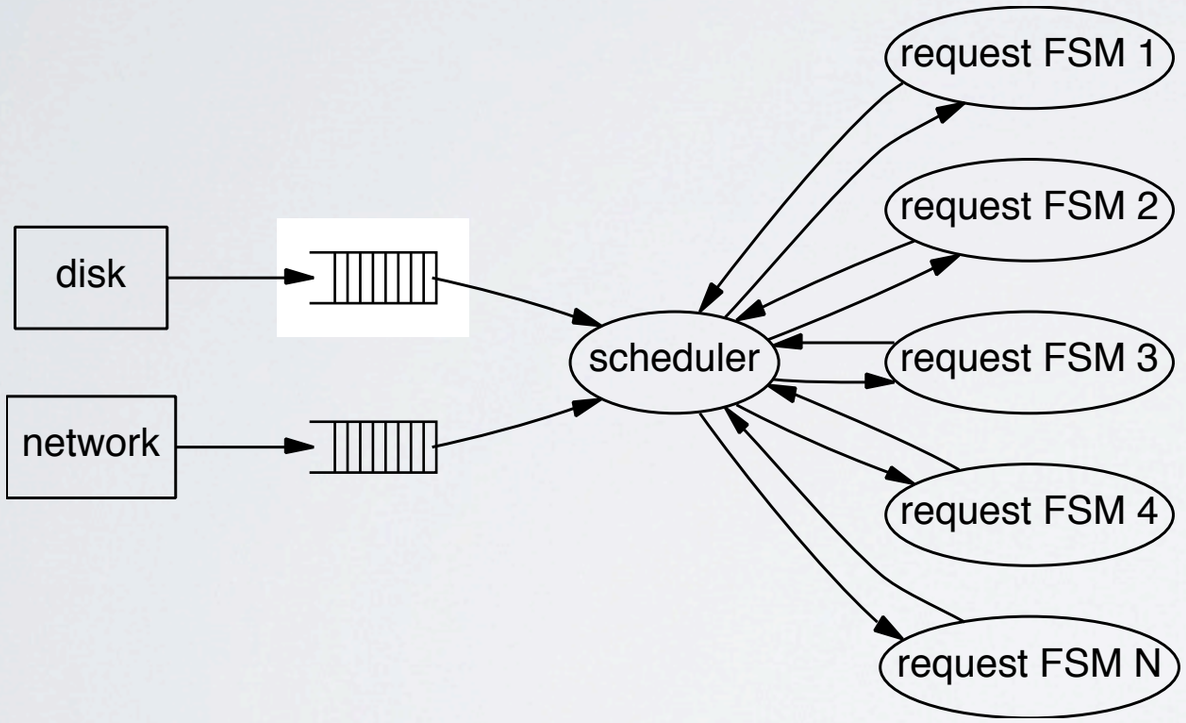
- threaded servers failed to scale up throughput
- event-based programming became popular
- threads are simpler and more natural
- downsides of threads due to poor implementations
- thread package + compiler support = right paradigm

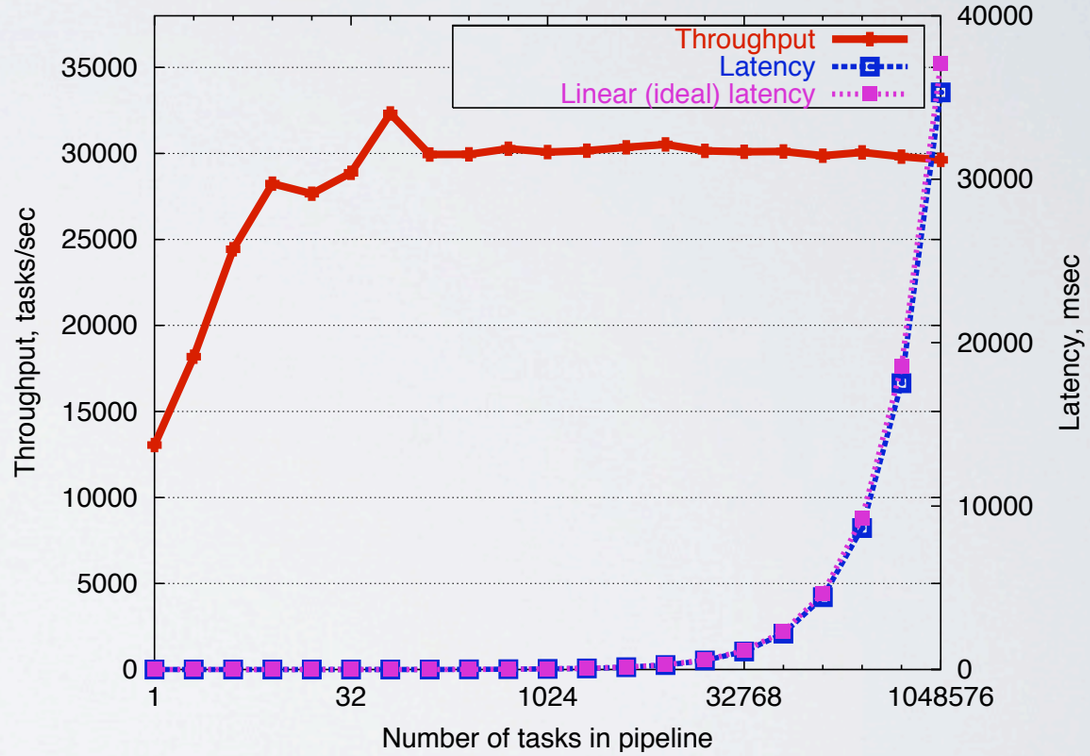
THREADS


request 1
request 2
request 3
request 4
request N
network
dispatcher
dispatch
send result
network


Throughput
Latency
Linear (ideal) latency
Throughput, tasks/sec
Latency, msec
Number of threads
30000
25000
20000
15000
10000
5000
0
400
350
300
250
200
150
100
50
0
1
4
16
64
256
1024

EVENTS


request FSM 1
request FSM 2
request FSM 3
request FSM 4
request FSM N
scheduler
disk
network


Throughput
Latency
Linear (ideal) latency
Throughput, tasks/sec
Latency, msec
Number of tasks in pipeline
35000
30000
25000
20000
15000
10000
5000
0
40000
30000
20000
10000
0
1
32
1024
32768
1048576

DUALITY

Events	Threads
event handlers	monitors
events accepted by handler	functions exported by module
SendMessage / AwaitReply	procedure call, or fork/join
SendReply	return from procedure
waiting for messages	waiting on condition variables

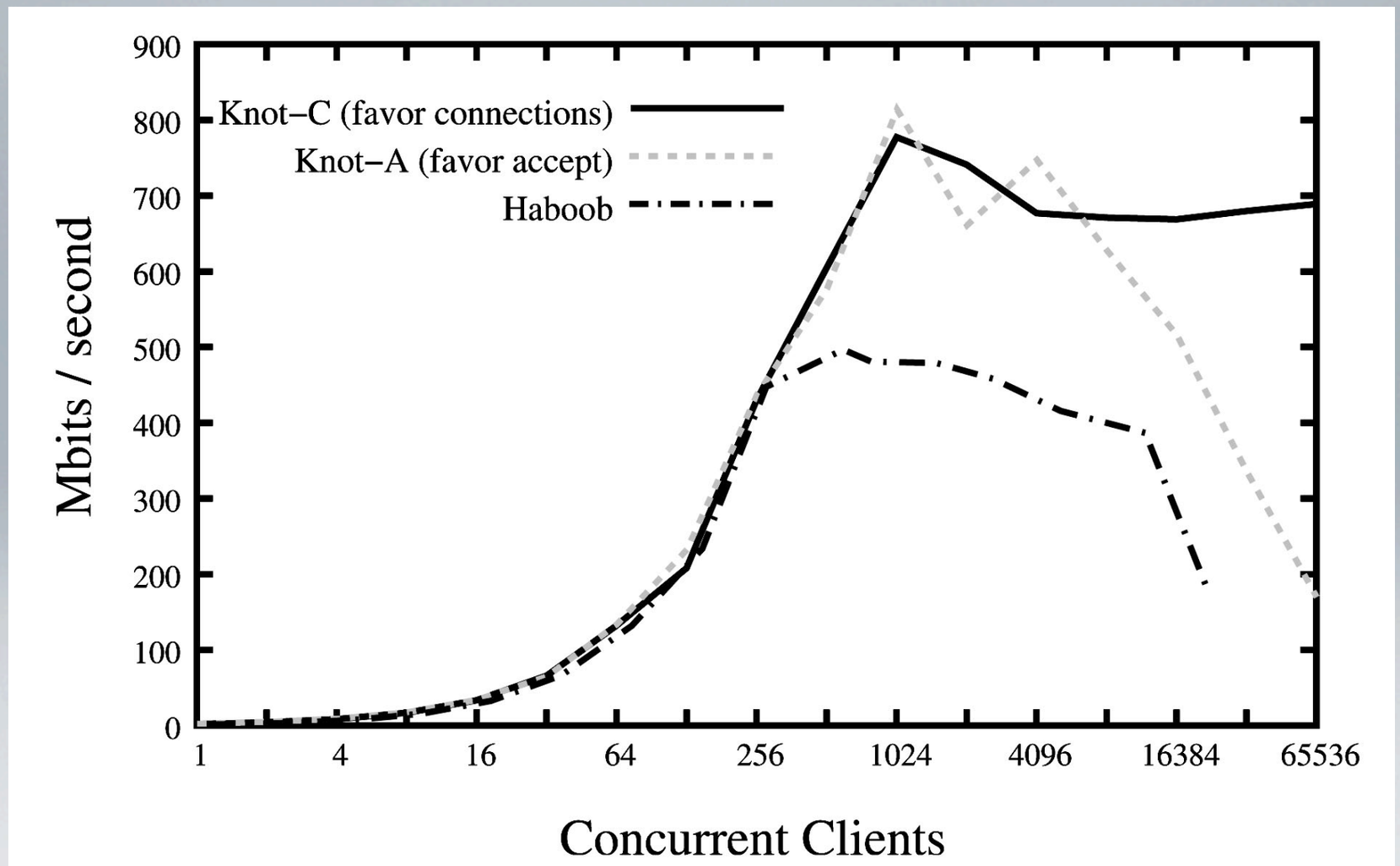
NON-PROBLEMS

Problem	Solution
performance	optimized user-level threading
restrictive control flow	complicated patterns not used
expensive synchronisation	cooperative threads
inefficient state management	stack-shrinking compiler
uninformed scheduling	magic?

AGAINST EVENTS

- events obfuscate control flow
- cluttered exception handling and state lifetime
- event-driven systems fall back to threads for complex parts
- fixing problems of events is worse than using threads

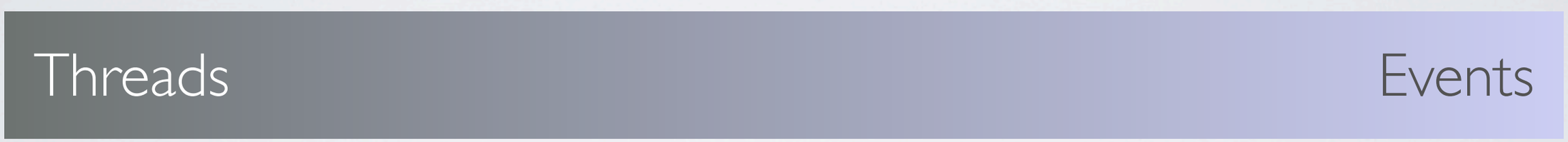
EVALUATION


Knot–C (favor connections)
Knot–A (favor accept)
Haboob
Mbits / second
900
800
700
600
500
400
300
200
100
0
1
4
16
64
256
1024
4096
16384
65536
Concurrent Clients

NON-PROBLEMS

Problem	Solution
performance	optimized user-level threading
restrictive control flow	complicated patterns not used
expensive synchronisation	cooperative threads
inefficient state management	stack-shrinking compiler
uninformed scheduling	magic?

SPECTRUM



DISCUSSION

- events obfuscate control flow: Should you worry?
- state lifetime: garbage collection
- compilers can help event systems: closures
- Fixing problems with threads turns them into events?
- Are desktop apps like servers? Or do they exhibit the complicated fan-in/fan-out patters the authors dismiss?