SEDA
-
An Architecture for Well-Conditioned Scalable Internet Services

Matt Welsh, David Culler and Eric Brewer,
University of California, Berkeley

Domaine et Enjeux

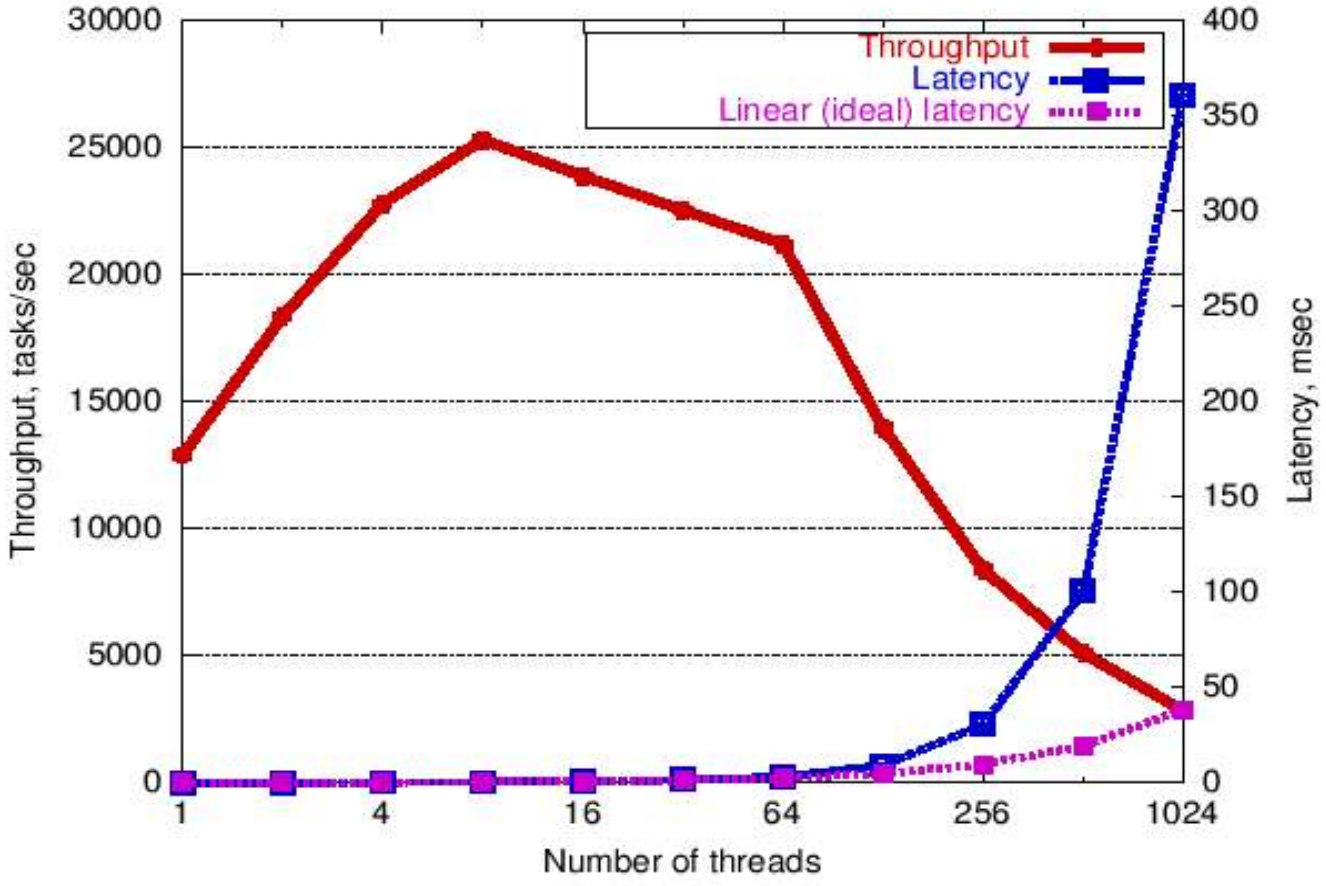
- conception de serveurs Internet
- conditions :
  - concurrence massive - avec plusieurs milliers de connections simultanées
  - fortes variations de charge (e.g nombre de demandes multiplié par 100 lorsqu'un service devient populaire)
- applications concernées :
  - de plus en plus variées (web services, peer-to-peer, etc.), contenu dynamique
  - on ne souhaite pas développer une plateforme spécifique à chaque fois

Métaphore du Tuyau

- « well-conditioned »
- « bien conditionné »
- métaphore du tuyau :
  - le débit augmente proportionnellement à la charge
  - jusqu'à un maximum (profondeur du tuyau) puis reste stable
  - quand le système est au débit maximum, la latence (temps d'attente) augmente linéairement avec le nombre de clients

Thread-Based Concurrency

- architecture de serveur la plus simple
- un thread pour chaque nouvelle requête

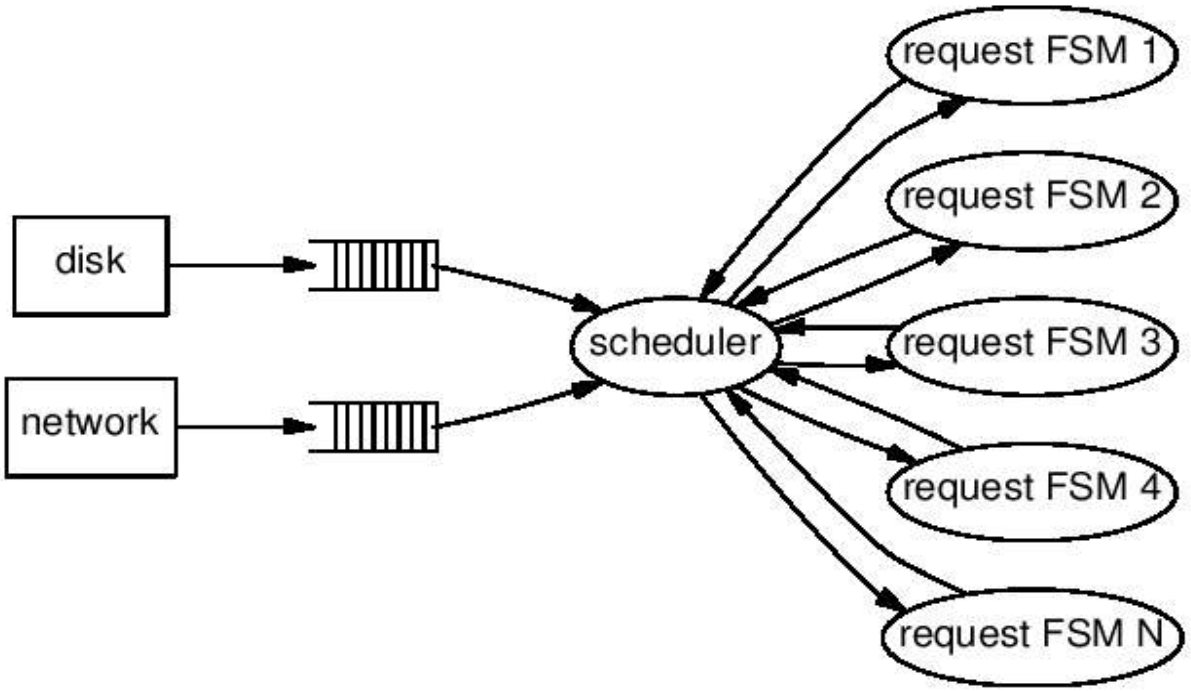


Bounded Thread Pools

- amélioration de la simple thread-based concurrency
- éviter la dégradation du débit
- utilise un pool de threads limité
- e.g Apache, IIs, WebSphere, etc.
- limite aussi le nombre de requêtes simultanées
- pas bien adapté à la concurrence massive
- pas très équitable (e.g cache hit vs. cache miss)

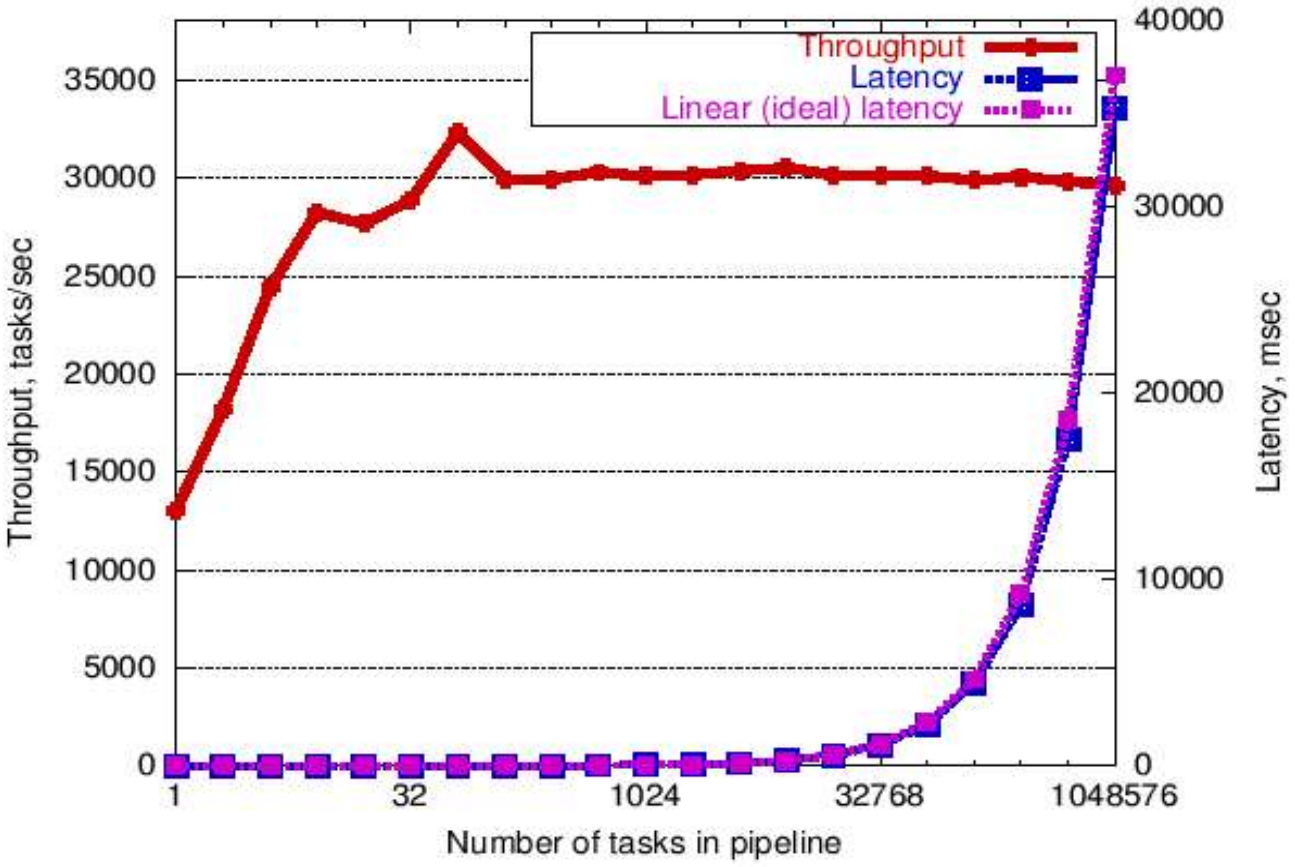
Event-Driven Concurrency

- essaie de dépasser les problèmes de passage à l'échelle des threads
- FSM : scheduler + event handlers



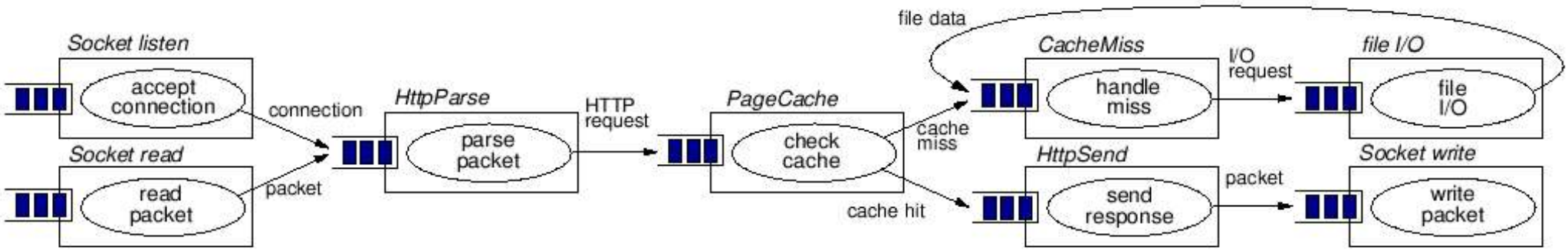
Event-Driven Concurrency

- proche du tuyau
- pas simple à développer (scheduler), trop spécifique, pas assez modulaire



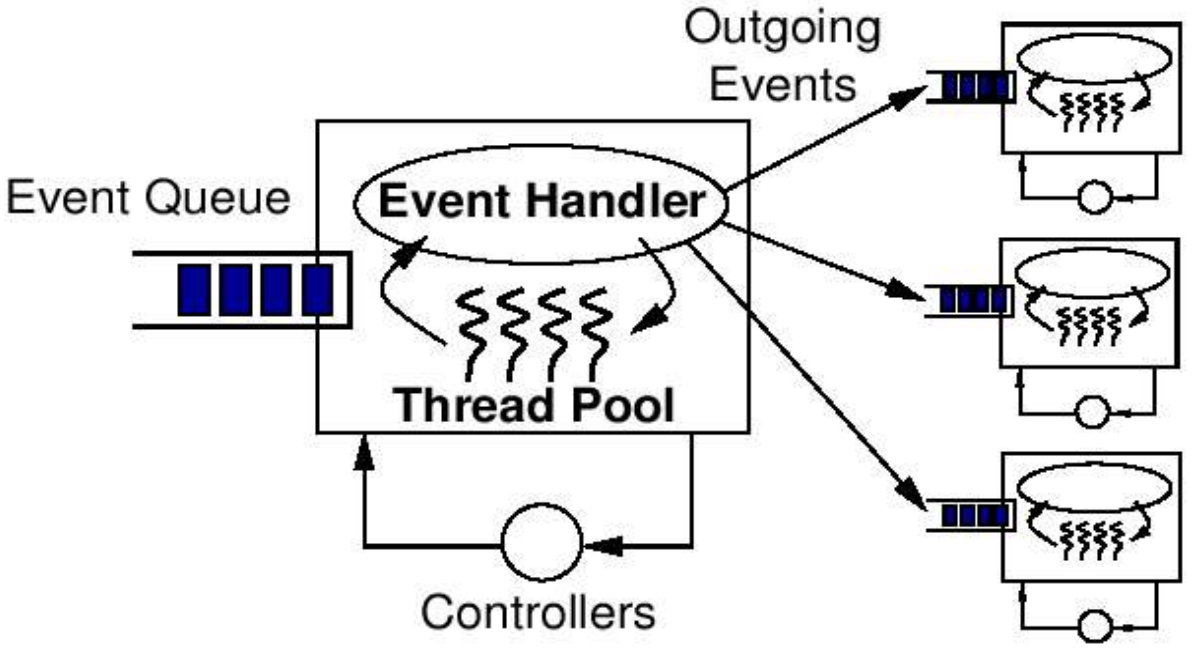
Staged Event-Driven

- propose de combiner les deux approches (threads et events)
- réseau d'étapes (stages) séparées par des files d'attente d'évenements (event queues)



Stage

- composant de base d'un service SEDA
- event handler
- event queue
- thread pool
- controller



Buts

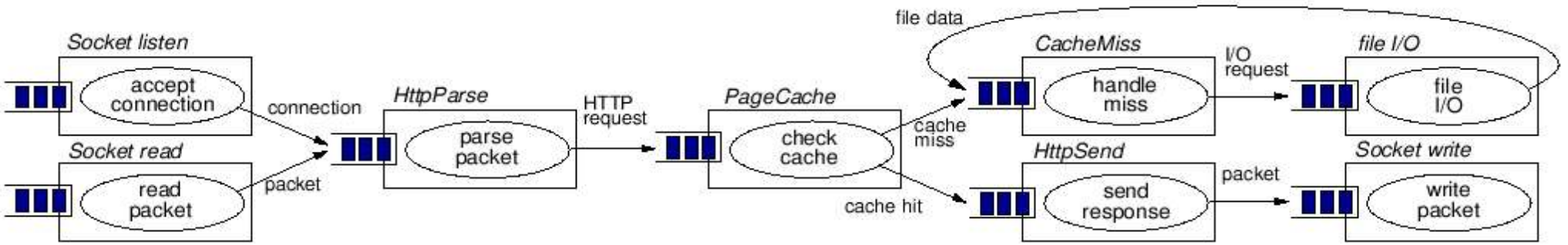
- supporter la concurrence massive
  - utilise l'approche event-driven pour éviter les dégradations dues au threads
- simplifier la construction de serveurs
  - propose une plateforme générique et modulaire
  - épargne aux développeurs une bonne partie de la complexité de l'ordonnancement et de la gestion des ressources
  - facilite le débogage et l'évaluation

Buts

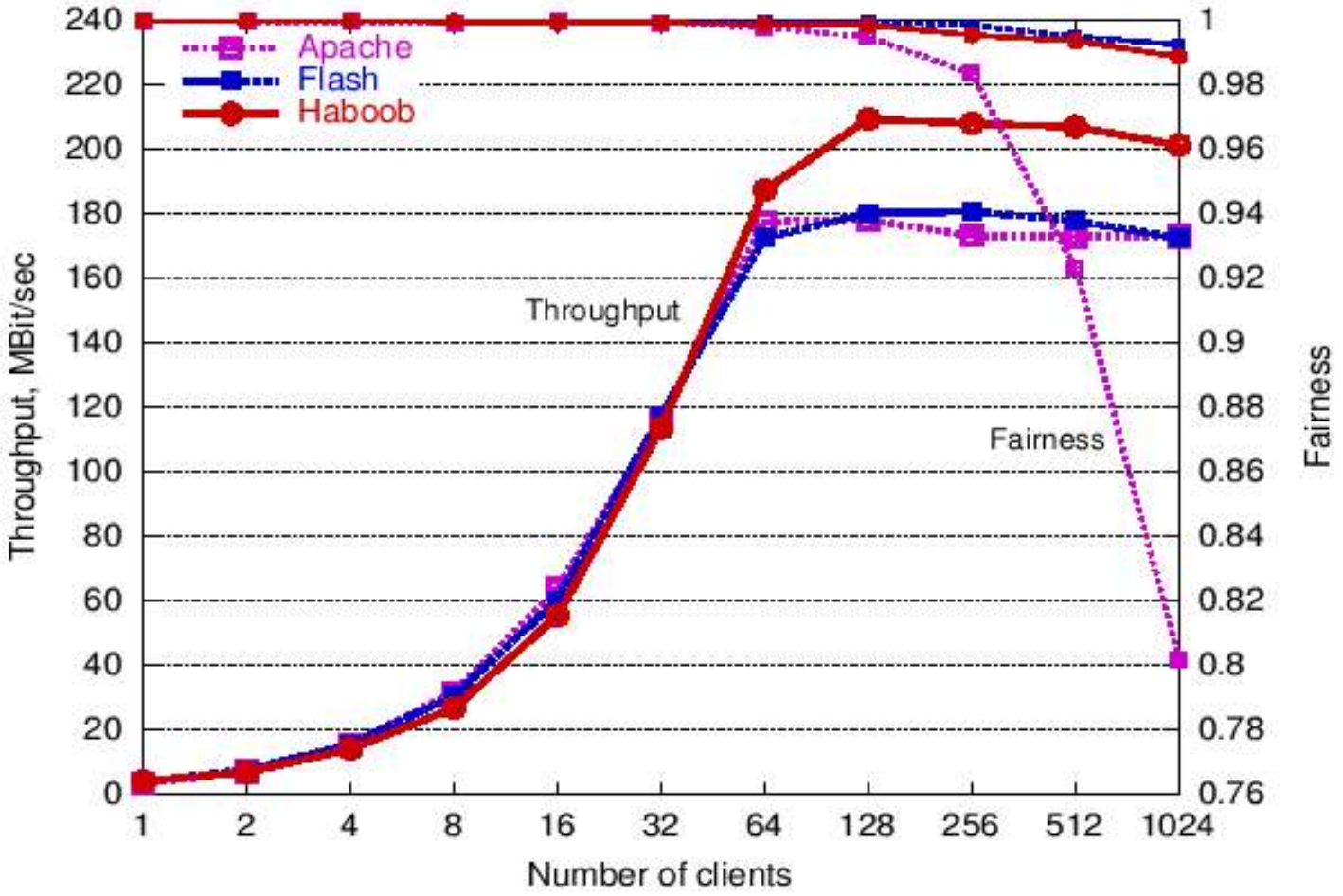
- permettre l'introspection
  - l'application peut analyser ses flux traités et adapter son comportement aux conditions de charge (e.g cache hit vs. cache miss)
  - possible dégradation élégante des services en cas de charge excessive
- proposer une gestion des ressources auto-adaptative
  - le système peut estimer les conditions de charge et s'y adapter
  - contrôleurs dynamique
  - (e.g nombre de threads, batch)

Haboob

- serveur HTTP selon SEDA
- construit avec le framework Sandstrom
- en java
- comparé à Apache (thread-based) et Flash (event-driven) selon le benchmark SPECweb99

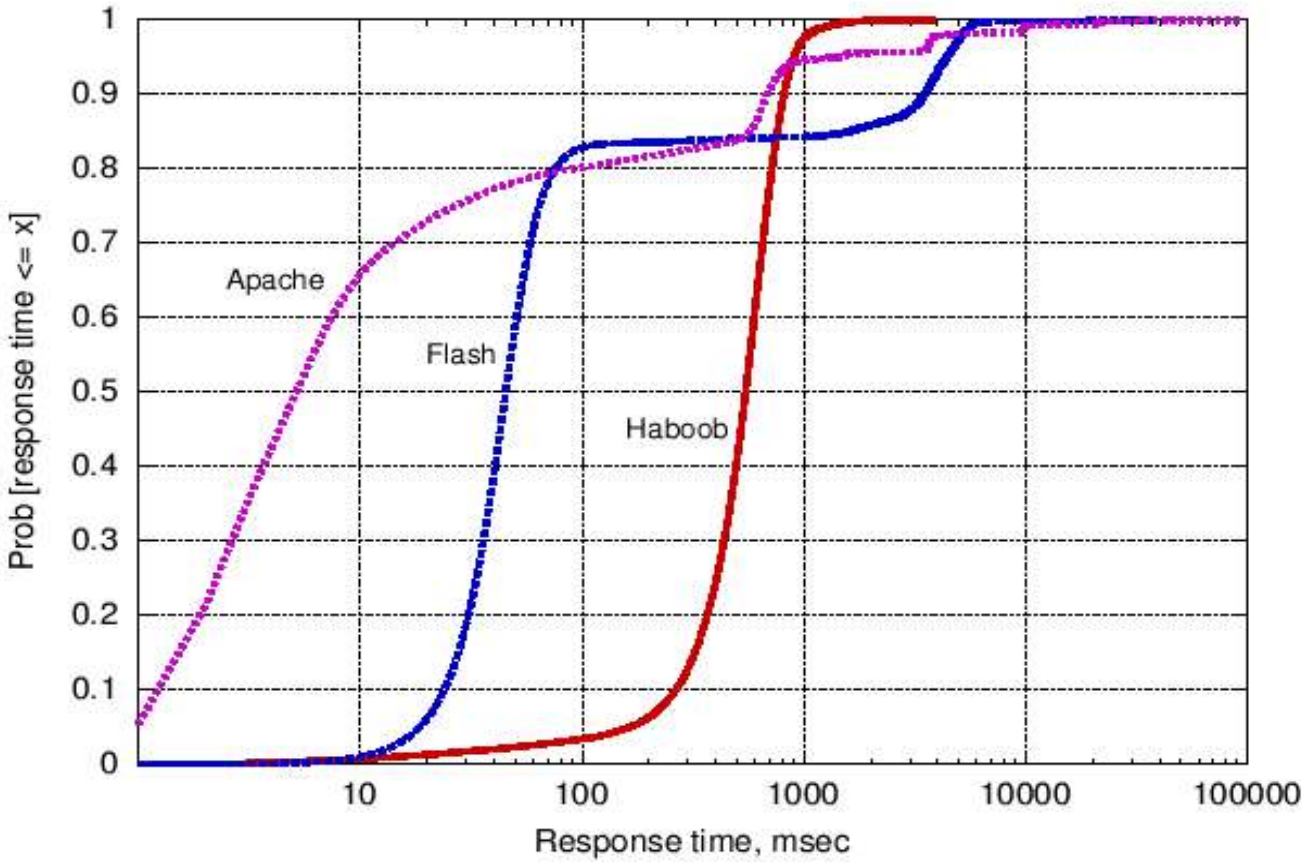


Throughput



- N.B Apache ne supporte que 150 connexions et Flash 506

Temps de Réponse



	SEDA	Flash	Apache
Mean RT	547 ms	665 ms	475 ms
Max RT	3.8 sec	37 sec	1.7 minutes

Réalisations

- Sandstrom, framework Java
- Haboob, web server
- Gnutella packet router
- Arashi, service mail
- OceanStore, stockage peer-to-peer

Conclusion

- nouveau paradigme de conception
- combine l'approche thread-based et l'approche event-driven et résoud chacun de leur problèmes :
  - performances pour les threads
  - ingénierie pour les évenements
- propose une approche mixte : performances / ingénierie et offre une solution simple, générique et efficace