A Self-Adapting Web Server Architecture:
Towards Higher Performance and Better Utilization

(Part I)

By: Khalid Al-Issa

Advisor
Dr. Farag Azzedin
Agenda

1. Objectives
2. What is Performance?
3. Improving Performance
4. Performance and I/O
5. Two Original Architectures
6. Need for New Ideas
7. Conclusion
Objectives

1. Define performance
2. Introduce I/O classes
3. Explain the two original approaches
4. Motivate new ideas
What is Performance?

- **Macro-Performance**
  - Throughput and response time

- **Micro-Performance**
  - Clock per Instruction (CPI)
  - Cache miss rate
  - I/O handling
  - Utilization of resources

Improving Performance

❖ Enhancing what users perceives
  ▪ Ex. Replication
  ▪ Pro’s: Simple & provides multiples of throughput
  ▪ Con’s: Issues continue to exist

❖ Enhancing internal operation
  ▪ Ex. Scheduling similar computations together
  ▪ Pro’s: Much more effective
  ▪ Con’s: More work, but is worth that

Performance and I/O

- **Job is to deliver contents**
  - Pages, images, scripts, database contents ... etc
  - Can be either **cached**, or **read from disk**

- **Can’t have everything in cache**
  - Disk I/O becomes a must
  - Concurrency is also a must
  - 4 classes of disk I/O

```
<table>
<thead>
<tr>
<th>Synch.</th>
<th>Blocking</th>
<th>Non-blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read/Write</td>
<td>Read/Write (O_NONBLOCK)</td>
</tr>
<tr>
<td>Asynch.</td>
<td>IO Multiplexing Select/Poll</td>
<td>AIO</td>
</tr>
</tbody>
</table>
```

Two Original Architectures

1) **Single-Process Event-Driven (SPED)**
   - **Analogy**
     - Single instance
     - Processing based on events
   - **Architecture layout**
Two Original Architectures

- **SPED Pro’s:**
  - Event-Driven is effective

- **Con’s:**
  - No mature Asynch. I/O built-in libraries
  - Too restrictive I/O scenarios
Two Original Architectures

2) Multi-Threaded and Multi-Process
   - Differences between the two
     - Sharing
     - Scheduling
   ✓ Analogy
   ✓ Architecture layout
Two Original Architectures

- **Multi-Threaded Pro’s**
  - Simple operation
  - Easy coding

- **Con’s:**
  - Low utilization (blocking IO, and no events)
  - Limited scalability
Need for New Ideas

- Serious limitations in original models
- New models should have:
  - Power and availability (Multi-Threaded)
  - Effectiveness and high utilization (Event-Driven)
Conclusion

❖ Concurrency is highly desirable
  ▪ Single process with Asynch. IO
    ▪ Limitation of Asynch. system calls
  ▪ Multiple instances of server (threads)
    ▪ Limited scalability and utilization

❖ There is a need for new models

❖ Next, we:
  ▪ Survey existing models
  ▪ Propose a new architecture