Abstract

The execution architecture determines largely the realtime and performance behavior of a system. Hard real time is characterized as ”missing a deadline” will result in system failure, while soft real time will result ”only” in dissatisfaction. An incremental design approach is described. Concepts such as latency, response time and throughput are illustrated. Design considerations and recommendations are given such as separation of concerns, understandability and granularity. The use of budgets for design and feedback is discussed.
Execution Architecture

Execution architecture concepts

version: 1.1
August 21, 2020
CVexecutionArchitecture
Fuzzy customer view on real time

- hard real time
- soft real time

Disastrous failure
Dissatisfaction

Human device safety
Waiting time

Loss of functionality
Limited throughput

Loss of information
Loss of eye hand coordination

Execution architecture concepts
Gerrit Muller

version: 1.1
August 21, 2020
EACHardVsSoft
**Smartening requirements**

Limited set of hard real time cases

- Precise form of the distribution is not important.
- Be aware of systematic effects

Well defined set of performance critical cases

- No exception allowed
- Worst case must fit

Typical within desired time, limited exceptions allowed.

Exceptions may not result in functional failure
Latency

connection latency

perceived delay

connection latency

long distance connection
Response Time

- total response time
- zap repetition
- visual feedback
- open for next response
- new channel
- visual feedback time

- remote control
- new channel
- zap

Execution architecture concepts
6 Gerrit Muller
throughput:
+ processing steps/frame
+ frames/second
+ concurrent streams
bus bandwidth, processor load [memory usage]
useful macroscopic views, be aware of microscopic behavior

- margin
- loss = not schedulable
- overhead
  - bus, OS, scheduling
- function 4
- function 3
- function 2
- function 1

depends on design
depends strongly on granularity

application overhead is still in this "nett" number
Design recommendations separation of concerns

- Separation of concerns
- Soft Real Time
- Hard Real Time
- HW
- HW
- HW
- Minimize influence
- Decoupling
- Minimal shared resources
- Queues or buffers
- Clear single demarcation between hard and soft
- Process as unit of execution
- Performance
- Separation
- Manage tension explicit
- Cost

Execution architecture concepts
Gerrit Muller
Design recommendations understandability

- Hard real time systems should be explainable with a few A4 diagrams.
- Overview is based on understanding many (critical) details.
- Complex reality; many details, many relations.
- Limited use of tasks, threads, priorities.
- Reasoning must be possible to combine or not to combine?
- Simulation: additional means if declared indispensable this is often a symptom of poor models.

Simple is better.
Granularity considerations

<table>
<thead>
<tr>
<th>unit of buffering</th>
<th>==</th>
<th>unit of synchronization</th>
<th>==</th>
<th>unit of processing</th>
<th>==</th>
<th>unit of I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>or</td>
<td>&lt;&gt;</td>
<td>or</td>
<td>&lt;&gt;</td>
<td>or</td>
<td>&lt;&gt;</td>
<td>or</td>
</tr>
</tbody>
</table>

- **video frame**
- **video line**
- **pixel**

*Fine grain:*
- flexible
- high overhead

*Coarse grain:*
- rigid
- low overhead
Design patterns

**synchronous**
- safety critical, reliable, subsystems
- very low overhead
- predictable
- understandable
- works best in total separation
- does not work for multiple rhythms

**thread based**
- Asynchronous applications and services
- separation of timing concerns
- sharing of resources (no wait)
- poor understanding of concurrency
- danger of high overhead

**timer based**
- regular rhythm;
- low "tunable" overhead
- understandable
- fast rhythms significant overhead

**interrupt based**
- I/O and HW events
- separation of timing concerns
- definition of interrupts determines:
  - overhead, understandability
Synchronous design

Execution architecture concepts

version: 1.1
August 21, 2020
EACsynchronousDesign
Actual timing on logarithmic scale

- **1 cycle 2 GHz CPU**
- **Pure context switch**
- **Zero message transfer**
- **Application level message exchange**
- **Application level function response**
- **100 Hz TV frame**
- **Human eye-hand co-ordination**
- **Human first irritation threshold**
- **Human second irritation threshold**

**Time Scale**
- **(ps) 10^{-12}**
- **(ns) 10^{-9}**
- **(μs) 10^{-6}**
- **(ms) 10^{-3}**
- **(s) 1**

**Processes**
- **FO4 inverter delay**
- **Cycle 2 GHz CPU**
- **DRAM cycle time**
- **DRAM latency**
- **1 byte transfer fast Ethernet**
- **1 package transfer fast Ethernet**
- **Disk seek**
- **Application level network message exchange**
- **Application level function response**

**From Low to High Level Processing Times**
Typical micro benchmarks for timing aspects

<table>
<thead>
<tr>
<th>Category</th>
<th>Infrequent operations, often time-intensive</th>
<th>Often repeated operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>start session, finish session</td>
<td>perform transaction, query</td>
</tr>
<tr>
<td>network, I/O</td>
<td>open connection, close connection</td>
<td>transfer data</td>
</tr>
<tr>
<td>high level construction</td>
<td>component creation, component destruction</td>
<td>method invocation</td>
</tr>
<tr>
<td>low level construction</td>
<td>object creation, object destruction</td>
<td>method invocation</td>
</tr>
<tr>
<td>basic programming</td>
<td>memory allocation, memory free</td>
<td>function call</td>
</tr>
<tr>
<td></td>
<td></td>
<td>loop overhead</td>
</tr>
<tr>
<td></td>
<td></td>
<td>basic operations (add, mul, load, store)</td>
</tr>
<tr>
<td>OS</td>
<td>task, thread creation</td>
<td>task switch, interrupt response</td>
</tr>
<tr>
<td>HW</td>
<td>power up, power down, boot</td>
<td>cache flush, low level data transfer</td>
</tr>
</tbody>
</table>
The transfer time as function of blocksize

\[
\text{time} = \text{block size} \times \text{rate}^{-1} + t_{\text{overhead}}
\]

worst case

optimal block-size
Example of a memory budget

<table>
<thead>
<tr>
<th>Service</th>
<th>Code</th>
<th>Obj Data</th>
<th>Bulk Data</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared code</td>
<td>11.0</td>
<td></td>
<td></td>
<td>11.0</td>
</tr>
<tr>
<td>User Interface process</td>
<td>0.3</td>
<td>3.0</td>
<td>12.0</td>
<td>15.3</td>
</tr>
<tr>
<td>Database server</td>
<td>0.3</td>
<td>3.2</td>
<td>3.0</td>
<td>6.5</td>
</tr>
<tr>
<td>Print server</td>
<td>0.3</td>
<td>1.2</td>
<td>9.0</td>
<td>10.5</td>
</tr>
<tr>
<td>Optical storage server</td>
<td>0.3</td>
<td>2.0</td>
<td>1.0</td>
<td>3.3</td>
</tr>
<tr>
<td>Communication server</td>
<td>0.3</td>
<td>2.0</td>
<td>4.0</td>
<td>6.3</td>
</tr>
<tr>
<td>UNIX commands</td>
<td>0.3</td>
<td>0.2</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Compute server</td>
<td>0.3</td>
<td>0.5</td>
<td>6.0</td>
<td>6.8</td>
</tr>
<tr>
<td>System monitor</td>
<td>0.3</td>
<td>0.5</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>Application SW total</td>
<td>13.4</td>
<td>12.6</td>
<td>35.0</td>
<td>61.0</td>
</tr>
<tr>
<td>UNIX Solaris 2.x</td>
<td></td>
<td></td>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td>File cache</td>
<td></td>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>74.0</td>
</tr>
</tbody>
</table>
### Complicating factors and measures

<table>
<thead>
<tr>
<th>complications</th>
<th>measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>considered margin</td>
</tr>
<tr>
<td>bus allocation</td>
<td>explicit behavior</td>
</tr>
<tr>
<td>memory management</td>
<td>architecture rules</td>
</tr>
<tr>
<td>garbage collection</td>
<td>monitoring, logging</td>
</tr>
<tr>
<td>memory (buffer, storage) fragmentation</td>
<td>pool management</td>
</tr>
<tr>
<td>non preemptable OS activities</td>
<td>feedback to architect</td>
</tr>
<tr>
<td>&quot;hidden&quot; dependencies (ie [dead]locks)</td>
<td>flipover simulation</td>
</tr>
<tr>
<td>systematic &quot;coincidences&quot;, avalanche triggers</td>
<td></td>
</tr>
<tr>
<td>instable response, performance</td>
<td></td>
</tr>
</tbody>
</table>