DISTRIBUTED OPERATING SYSTEMS
SCALABILITY AND NAMING

HORST SCHIRMEIER, DISTRIBUTED OPERATING SYSTEMS, SS2024
Lecturer in charge of DOS: Dr. Carsten Weinhold, Barkhausen Institute TUD

Several lectures presented by research-group members

Mandatory: register for mailing list (see website)
  - must use “tu-dresden.de” mail addresses
  - Hybrid format (BBB, recordings, both “best effort”)
    - Lecture: Monday, 11:10
    - Exercise: Monday 13:00
      (roughly every 2 weeks, starting 2024-04-15)
EXAMS

- Oral exam covering lectures and exercises
- About 1 exam date per month
- Exam appointments:
  - Email to sandy.seifarth-haupold@tu-dresden.de
  - Provide paperwork (forms) at least 2 weeks before exam otherwise, automatic cancellation (and angry secretary)
  - You can cancel until 2 weeks before date; after that, no more cancellation except for sickness.
- Diplom/Master INF study programmes:
  - can be combined with other classes in complex modules
• **Course name** no more precise, rather: “Interesting/advanced Topics in Operating Systems”
  - Scalability
  - Systems security
  - Modeling
  - Some overlap with “Distributed Systems” (Dr. Springer / Prof. Wählsch) and some classes by Prof. Fetzer
  - In some cases no written material (except slides)
1.0) DOS ORGANISATION

1.1) SCALABILITY IN COMPUTER SYSTEMS

1.2) EXAMPLE: DNS/BIND
GOAL OF ALL LECTURES ON SCALABILITY

Topics:
- Scalability: terminology, problems, principle approaches
- Case studies, all layers of compute systems

Goal:
- Understand (some of the) important principles how to build scalable systems
Outline:

- **Scalability** – and a simple model to reason about one aspect
- **Names** in Distributed Systems: purposes of naming, terminology (DNS)
- **Application of scalability** approaches on name resolution

Goal:

- Understand some of the important principles how to build scalable systems (using DNS as example)
- Memory consistency
- Locks and advanced synchronization approaches
- File systems
- Load balancing (MosiX) and HPC (MPI)
Scalability:

Scalability is the property of a system to handle a growing amount of work by adding resources to the system.

(Wikipedia (2019) and many other sources)
SCALABILITY: WEAK ././ STRONG

Ability of a system to use growing resources ...

- **Weak scalability:**
  to handle growing load, larger problem, ...

- **Strong scalability:**
  accelerate existing work load, same problem
PROBLEMS

- Performance bottlenecks / Amdahl's Law
- Failures / abuse
- Administration
RESOURCES AND PERFORMANCE

- Processors
- Communication
- Memory (remember basic OS course: “thrashing”)

Speedup: \[
\frac{\text{original execution time}}{\text{enhanced execution time}}
\]
SIMPLE MODEL: AMDAHL’S LAW

\[
\text{Speedup} = \frac{\text{original execution time}}{\text{enhanced execution time}}
\]

Parallel Execution

- **red**: cannot run in parallel
- **green**: runs perfectly parallel
- unlimited processors maximum speedup: **blue/red**
Parallel Execution, N processors

red: cannot run in parallel

green: runs perfectly parallel

N processors maximum speedup: blue/(red+green/N)
AMDAHL’S LAW

Parallel Execution, N processors

red: cannot run in parallel

green: runs perfectly parallel

maximum speedup: blue/(red+green/N)
AMDAHL’S LAW

\[ \text{Speedup} = \frac{\text{original execution time}}{\text{enhanced execution time}} \]

- P: section that can be parallelized
- 1-P: serial section
- N: number of CPUs

\[ \text{Speedup}(P,N) = \frac{1}{1 - P + \frac{P}{N}} \]

- if N becomes VERY large, speedup approaches: 1/(1-P)
THE “RPC” PRINCIPLES

- **Partitioning**
  Split systems into parts that can operate independently/parallel to a large extent

- **Replication**
  Provide several copies of components
  - that are kept consistent eventually
  - that can be used in case of failure of copies

- **Locality** (caching)
  Maintain a copy of information that is nearer, cheaper/faster to access than the original
MORE PRINCIPLES

- Identify and address **bottlenecks**
- **Specialize** functionality/interfaces
- Right level of **consistency**
  Caches, replicates, ... need not always be fully consistent.
- Lazy information dissemination
- Balance load (make partitioning dynamic)
1.0) DOS ORGANISATION
1.1) SCALABILITY IN COMPUTER SYSTEMS
1.2) EXAMPLE: DNS/BIND
UUCP/MMDF:

- `ira!gmdzi!oldenburg!heinrich!user` (path to destination)
- `user@ira!heinrich%gmdzi`
  (mixing identifiers and path information)
ARPA-Net at the beginning:
- a single file: hosts.txt
- maintained at Network Information Center of SRI (Stanford)
- accessed via FTP
- TCP/IP in BSD Unix massively increased ARPA-Net size
  → Chaos, name collisions, consistency, load, ...

**DNS**: Paul Mockapetris et al.
DOMAIN NAME SYSTEM

hsc@os.inf.tu-dresden.de
NAMES, IDENTIFIERS, ADDRESSES

- **Names**
  - symbolic, many names possible for one entity
  - have a *meaning for people*

- **Identifiers**
  - identifies an entity *uniquely*
  - are used by programs

- **Addresses**
  - *locates* an entity
  - changes occasionally (or frequently)
- **Name resolution:**
  Map symbolic names to a set of attributes such as: identifiers, addresses, alias names, security properties, encryption keys, ...

- **Principle interface:**
  - **Register** (Context, Name, attributes, ...)
  - **Lookup** (Context, Name) → attributes
**DNS DOMAINS**

**Domain** = subtree in DNS hierarchy:
- **de**
- **tu-dresden.de**
- **os.inf.tu-dresden.de**
- **tudos.org** and **os.inf.tu-dresden.de** are aliases

**DNS DOMAINS**

- **org**
- **edu**
- **com**
- **us**
- **au**
- **de**
- **wikipedia**
- **tudos**
- **tu-dresden**
- **dresden**
- **zih**
- **inf**
- **fis**
- **os**
- **xy..**
- **can..**
- **erwin**
• **Zone**: Subset of a domain over which an **authority** has complete control → controlled by a **name server**

• **Subzones** can be delegated to other authorities.

• **Navigation**: querying in a set of cooperating name servers
POTENTIAL ZONES

de

- tu-dresden
  - zih
  - inf
  - os
  - xy..

- dresden
  - fis
  - can..
  - erwin
Option #1: complete tu-dresden domain
- Option #1: complete tu-dresden domain
- Option #2: Opt. #1 with sub zone **os** (not allowed by ZIH anymore)
CACHING

- remember intermediate results
- @ root NS makes no sense! (overload)
- @ NS i!
CACHING

Scalability in Computer Systems, Example: DNS/BIND
Two techniques for replication:

- Several IPs/names
- "anycast" (send packet to one of many servers with same IP)

- 13 root name server IPs, ~1700 physical servers via anycast

- Each zone has at least one primary and one secondary IP
## RESOURCE RECORDS

<table>
<thead>
<tr>
<th>Record type</th>
<th>Interpretation</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>address</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>AAAA</td>
<td>address</td>
<td>IPv6 address</td>
</tr>
<tr>
<td>NS</td>
<td>Name server</td>
<td>DNS name</td>
</tr>
<tr>
<td>CNAME</td>
<td>Symbolic link</td>
<td>DNS name of canonical name</td>
</tr>
<tr>
<td>SOA</td>
<td>Start of authority</td>
<td>Zone-specific properties</td>
</tr>
<tr>
<td>PTR</td>
<td>IP reverse pointer</td>
<td>DNS name</td>
</tr>
<tr>
<td>HINFO</td>
<td>Host info</td>
<td>Text description of host OS</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

name =>
SUMMARY

- Main problems for scalability
- Simple model: Amdahl’s law
- Few principle approaches
- DNS as fine example, more to come
  → study DNS it in your first exercise (Apr 15th)

- Register in mailing list!
  (with a tu-dresden.de address)

• Mark Hill, Michael Marty: **Amdahl's Law in the Multicore Era**, 2008 IEEE ([available online via SLUB](#))

• Couluris, Tollimore, Kindberg: **Distributed systems**