- 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, but **NO recordings**)
  - Lecture: Monday, 11:10
  - Exercise: **Monday 13:00** | | **Tuesday, 09:20**?
    (roughly every 2 weeks)
https://tinyurl.com/bdznzyvc
(→ terminplaner4.dfn.de)
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“ (Prof. Schill) 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

HORST SCHIRMEIER, DISTRIBUTED OPERATING SYSTEMS, SS2023
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)
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”)

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

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
EARLY EMAILS

- UUCP/MMDF:
  - `ira!gmdzi!oldenburg!heinrich!user` (path to destination)
  - `user@ira!heinrich@gmdzi`
    (mixing identifiers and path information)
A BIT OF HISTORY

- 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

org  edu  com  us  au  de  ch  at  x

wikipedia  todos  tu-dresden  dresden  amazon  bahn

zih  inf  fis  os  xy..  can..  erwin

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

- **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
Domain =
subtree in DNS hierarchy:

- de
- tu-dresden.de
- os.inf.tu-dresden.de
- tudos.org and os.inf.tu-dresden.de are aliases
PARTITIONING: ZONE

- **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
- 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

root NS

NS2

NS3

NS i

...
RECURSIVE ./.. ITERATIVE

 TU Dresden, Horst Schirmeier, Distributed Operating Systems, SS2023
 Scalability in Computer Systems, Example: DNS/BIND
Scalability in Computer Systems, Example: DNS/BIND

- **REPLICATION**

The diagram illustrates the replication structure of a DNS (Domain Name System) setup, showcasing how DNS servers are replicated across various nodes to enhance scalability. The root NS (Name Server) is replicated across multiple nodes, including NS1, NS2, and NS3, ensuring that queries can be resolved efficiently across the network.
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
<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>
- 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 17\textsuperscript{th}/18\textsuperscript{th})

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


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

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