WIKIPEDIA

Scalability

Scalability is the property of a system to handle a growing amount of work by adding resources to the system.[1]

In an economic context, a scalable business model implies that a company can increase sales given increased resources. For example, a package delivery system is scalable because more packages can be delivered by adding more delivery vehicles. However, if all packages had to first pass through a single warehouse for sorting, the system would not be scalable, because one warehouse can handle only a limited number of packages.[2]

In computing, scalability is a characteristic of computers, networks, algorithms, networking protocols, programs and applications. An example is a search engine, which must support increasing numbers of users, and the number of topics it indexes.[3] **Webscale** is a computer architectural approach that brings the capabilities of large-scale cloud computing companies into enterprise data centers.[4]

In mathematics, scalability mostly refers to closure under scalar multiplication.

Contents

Examples

The Incident Command System (ICS) is used by emergency response agencies in the United States. ICS can scale resource coordination from a single-engine roadside brushfire to an interstate wildfire. The first resource on scene establishes command, with authority to order resources and delegate responsibility (managing five to seven officers, who will again delegate to up to seven, and on as the incident grows). As an incident expands, more senior officers assume command.[5]

Dimensions

Scalability can be measured over multiple dimensions, such as:[6]

- *Administrative scalability*: The ability for an increasing number of organizations or users to access a system.
- *Functional scalability*: The ability to enhance the system by adding new functionality without disrupting existing activities.
- *Geographic scalability*: The ability to maintain effectiveness during expansion from a local area to a larger region.
- *Load scalability*: The ability for a distributed system to expand and contract to accommodate heavier or lighter loads, including, the ease with which a system or component can be modified, added, or removed, to accommodate changing loads.
- *Generation scalability*: The ability of a system to scale by adopting new generations of components.
- *Heterogeneous scalability* is the ability to adopt components from different vendors.

Domains

- A routing protocol is considered scalable with respect to network size, if the size of the necessary routing table on each node grows as $O(\log N)$, where N is the number of nodes in the network. Some early peer-to-peer (P2P) implementations of Gnutella had scaling issues. Each node query flooded its requests to all nodes. The demand on each peer increased in proportion to the total number of peers, quickly overrunning their capacity. Other P2P systems like BitTorrent scale well because the demand on each peer is independent of the number of peers. Nothing is centralized, so the system can expand indefinitely without any resources other than the peers themselves.
- A scalable online transaction processing system or database management system is one that can be upgraded to process more transactions by adding new processors, devices and storage, and which can be upgraded easily and transparently without shutting it down.
- The distributed nature of the Domain Name System allows it to work efficiently, serving billions of hosts on the worldwide Internet.

Horizontal (scale out) and vertical scaling (scale up)

Resources fall into two broad categories: horizontal and vertical.[7]

Horizontal or scale out

Scaling horizontally (out/in) means adding more nodes to (or removing nodes from) a system, such as adding a new computer to a distributed software application. An example might involve scaling out from one web server to three. High-performance computing applications, such as seismic analysis and biotechnology, scale workloads horizontally to support tasks that once would have required expensive supercomputers. Other workloads, such as large social networks, exceed the capacity of the largest supercomputer and can only be handled by scalable systems. Exploiting this scalability requires software for efficient resource management and maintenance.[6]

Vertical or scale up

Scaling vertically (up/down) means adding resources to (or removing resources from) a single node, typically involving the addition of CPUs, memory or storage to a single computer.[6]

Larger numbers of elements increases management complexity, more sophisticated programming to allocate tasks among resources and handle issues such as throughput and latency across nodes, while some applications do not scale horizontally.

Network function virtualization defines these terms differently: scaling out/in is the ability to scale by adding/removing resource instances (e.g., virtual machine), whereas scaling up/down is the ability to scale by changing allocated resources (e.g., memory/CPU/storage capacity).[8]

Database scalability

Scalability for databases requires that the database system be able to perform additional work given greater hardware resources, such as additional servers, processors, memory and storage. Workloads have continued to grow and demands on databases have followed suit.

Algorithmic innovations have include row-level locking and table and index partitioning. Architectural innovations include shared-nothing and shared-everything architectures for managing multi-server configurations.

Strong versus eventual consistency (storage)

In the context of scale-out data storage, scalability is defined as the maximum storage cluster size which guarantees full data consistency, meaning there is only ever one valid version of stored data in the whole cluster, independently from the number of redundant physical data copies. Clusters which provide "lazy" redundancy by updating copies in an asynchronous fashion are called 'eventually consistent'. This type of scale-out design is suitable when availability and responsiveness are rated higher than consistency, which is true for many web file-hosting services or web caches (*if you want the latest version, wait some seconds for it to propagate*). For all classical transaction-oriented applications, this design should be avoided.[9]

Many open-source and even commercial scale-out storage clusters, especially those built on top of standard PC hardware and networks, provide eventual consistency only. Idem some NoSQL databases like CouchDB and others mentioned above. Write operations invalidate other copies, but often don't wait for their acknowledgements. Read operations typically don't check every redundant copy prior to answering, potentially missing the preceding write operation. The large amount of metadata signal traffic would require specialized hardware and short distances to be handled with acceptable performance (i.e., act like a non-clustered storage device or database).

Whenever strong data consistency is expected, look for these indicators:

- the use of InfiniBand, Fibrechannel or similar low-latency networks to avoid performance degradation with increasing cluster size and number of redundant copies.
- short cable lengths and limited physical extent, avoiding signal runtime performance degradation.
- majority / quorum mechanisms to guarantee data consistency whenever parts of the cluster become inaccessible.

Indicators for eventually consistent designs (not suitable for transactional applications!) are:

- write performance increases linearly with the number of connected devices in the cluster.
- while the storage cluster is partitioned, all parts remain responsive. There is a risk of conflicting updates.

Performance tuning versus hardware scalability

It is often advised to focus system design on hardware scalability rather than on capacity. It is typically cheaper to add a new node to a system in order to achieve improved performance than to partake in performance tuning to improve the capacity that each node can handle. But this approach can have diminishing returns (as discussed in performance engineering). For example: suppose 70% of a program can be sped up if parallelized and run on multiple CPUs instead of one. If α is the fraction of a calculation that is sequential, and $1-\alpha$ is the fraction that can be parallelized, the maximum speedup that can be achieved by using P processors is given according to Amdahl's Law:

$$\frac{1}{\alpha + \frac{1-\alpha}{P}}.$$

Substituting the value for this example, using 4 processors gives

$$\frac{1}{0.3 + \frac{1-0.3}{4}} = 2.105.$$

Doubling the computing power to 8 processors gives

$$\frac{1}{0.3 + \frac{1-0.3}{8}} = 2.581.$$

Doubling the processing power has only sped up the process by roughly one-fifth. If the whole problem was parallelizable, the speed would also double. Therefore, throwing in more hardware is not necessarily the optimal approach.

Weak versus strong scaling

High performance computing has two common notions of scalability:

- *Strong scaling* is defined as how the solution time varies with the number of processors for a fixed *total* problem size.
- *Weak scaling* is defined as how the solution time varies with the number of processors for a fixed problem size *per processor*.[10]

See also

- Computational complexity theory
- Extensibility
- Gustafson's law
- List of system quality attributes
- Load balancing (computing)
- Lock (computer science)
- NoSQL
- Scalable Video Coding (SVC)
- Similitude (model)

References

1. Bondi, André B. (2000). *Characteristics of scalability and their impact on performance*. Proceedings of the second international workshop on Software and performance – WOSP '00.

p. 195. doi:10.1145/350391.350432 (https://doi.org/10.1145%2F350391.350432). ISBN 158113195X.

2. Hill, Mark D. (1990). "What is scalability?" (http://digital.library.wisc.edu/1793/9676). *ACM SIGARCH Computer Architecture News*. **18** (4): 18. doi:10.1145/121973.121975 (https://doi.org/10.1145%2F121973.121975). S2CID 1232925 (https://api.semanticscholar.org/CorpusID:1232925). and Duboc, Leticia; Rosenblum, David S.; Wicks, Tony (2006). *A framework for modelling and analysis of software systems scalability* (http://discovery.ucl.ac.uk/4990/1/4990.pdf) (PDF). Proceedings of the 28th international conference on Software engineering – ICSE '06. p. 949. doi:10.1145/1134285.1134460 (https://doi.org/10.1145%2F1134285.1134460). ISBN 1595933751.
3. Laudon, Kenneth Craig; Traver, Carol Guercio (2008). *E-commerce: Business, Technology, Society* (https://books.google.com/books?id=n4bUGAAACAAJ). Pearson Prentice Hall/Pearson Education. ISBN 9780136006459.
4. "Why web-scale is the future" (https://www.networkworld.com/article/3199205/why-web-scale-is-the-future.html). *Network World*. 2020-02-13. Retrieved 2017-06-01.
5. Bigley, Gregory A.; Roberts, Karlene H. (2001-12-01). "The Incident Command System: High-Reliability Organizing for Complex and Volatile Task Environments". *Academy of Management Journal*. **44** (6): 1281–1299. doi:10.5465/3069401 (https://doi.org/10.5465%2F3069401). ISSN 0001-4273 (https://www.worldcat.org/issn/0001-4273).
6. Hesham El-Rewini and Mostafa Abd-El-Barr (April 2005). *Advanced Computer Architecture and Parallel Processing* (https://books.google.com/books?id=7JB-u6D5Q7kC&q=parallel+architectures+scalability&pg=PA63). John Wiley & Sons. p. 66. ISBN 978-0-471-47839-3.
7. Michael, Maged; Moreira, Jose E.; Shiloach, Doron; Wisniewski, Robert W. (March 26, 2007). *Scale-up x Scale-out: A Case Study using Nutch/Lucene*. 2007 IEEE International Parallel and Distributed Processing Symposium. p. 1. doi:10.1109/IPDPS.2007.370631 (https://doi.org/10.1109%2FIPDPS.2007.370631). ISBN 978-1-4244-0909-9.
8. "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV" (http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV003v010201p.pdf) (PDF).
9. Sadek Drobi (January 11, 2008). "Eventual consistency by Werner Vogels" (http://www.infoq.com/news/2008/01/consistency-vs-availability). InfoQ. Retrieved April 8, 2017.
10. "The Weak Scaling of DL_POLY 3" (https://web.archive.org/web/20140307224104/http://www.stfc.ac.uk/cse/25052.aspx). STFC Computational Science and Engineering Department. Archived from the original (http://www.stfc.ac.uk/cse/25052.aspx) on March 7, 2014. Retrieved March 8, 2014.

External links

- Architecture of a Highly Scalable NIO-Based Server (http://today.java.net/pub/a/today/2007/02/13/architecture-of-highly-scalable-nio-server.html) – an article about writing scalable server in Java (java.net).
- Links to diverse learning resources (https://code.google.com/p/memcached/wiki/HowToLearnMoreScalability) – page curated by the memcached project.
- Scalable Definition (http://www.linfo.org/scalable.html) – by The Linux Information Project (LINFO)
- Scale in Distributed Systems (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.3576) B. Clifford Neumann, In: *Readings in Distributed Computing Systems*, IEEE Computer Society Press, 1994