3-бөлім Информатика

Раздел 3 Информатика

Section 3 Computer science

UDC 519.872

Study of enterprise load balancing algorithms using model-based design

Aidarov K.A., PhD student, al-Farabi Kazakh National University, Almaty, Republic of Kazakhstan, +77013460160, E-mail: kanataidarov@yahoo.com
Almatov A.Zh., Student, al-Farabi Kazakh National University, Almaty, Republic of Kazakhstan, +77476273606, E-mail: abat_aktobe_95@mail.ru

Given work describes load balancing algorithms for external services with unspecified clients used in real enterprise facilities. Simplest example of such service is a pair of a web server and a browser. Principles of load balancing, their pros and cons are considered, their implementation and applicability to different services are described. Algorithms of load balancing for DNS Round Robin, Direct Routing, Redirect methods are studied. Result of model implementation for cluster system allowed to obtain estimations of oscillation effect decrease between its overloaded and normal states. As a conclusion recommendations for using implemented model of queuing network for different enterprise services of user request processing are given.
Key words: load balancing, discrete-event simulation, model based design, probabilistic modeling.

Модельге бағытталған жобалау арқылы жүктеуді безбеңдеудің өндірістік алгоритмдерін зерттеу

Айдаров К.А., докторант, әл-Фараби атындағы Қазақ Ұлттық университеті, Алматы қ., Қазақстан Республикасы, +77013460160, E-mail: kanataidarov@yahoo.com
Алматов А.Ж., студент, әл-Фараби атындағы Қазақ Ұлттық университеті, Алматы қ., Қазақстан Республикасы, +77476273606, E-mail: abat_aktobe_95@mail.ru

Берілген жұмыс, шынайы өндірістік мекемелерде қолданылатын, арнайы емес клиенттері бар сырттай қызметтер үшін безбеңдеу алгоритмдерін сипаттайды. Бұндай қызметтің ең қарапайым мысалы ретінде веб-серверді немесе браузерді алуға болады. Безбеңдеу принциптері, олардың артықшылықтары мен кемшіліктері қарастырылған, түрлі қызметтер үшін іске асырылулары мен қолданымдылығы сипатталған. DNS Round Robin, Direct Routing, редирект үдістері үшін безбеңдеу алгоритмдері зерттелген. Модельді кластерлік жүйе үшін іске асыру нәтижесі бойынша, жүйенің асыра жүктелген және қалыпты жағдайлары арасындағы осцилляция әсерінің кемуі бакыланды. Макаланың соңында іске асырылған желілік жүйенің моделін, түрлі қолданушылардың сұраныстарын өңдеу қызметтері үшін, қолдануға нұсқаулықтар келтірілген.
Түйін сөздер: жүктеуді безбеңдеу, дискретті-оқиғалы модельдеу, модельге бағытталған жобалау, ықтималдық модельдеу

Исследование промышленных алгоритмов балансировки нагрузки через модельно-ориентированное проектирование

Айдаров К.А., докторант, Казахский национальный университет им. аль-Фараби, г. Алматы, Республика Казахстан, +77013460160, E-mail: kanataidarov@yahoo.com
Алматов А.Ж., студент, Казахский национальный университет им. аль-Фараби, г. Алматы, Республика Казахстан, +77476273606, E-mail: abat_aktobe_95@mail.ru

Данная работа описывает алгоритмы балансировки для внешних сервисов с неспециализированными клиентами, используемых в настоящих промышленных ЦОДах. Простейший пример такого сервиса и клиента – это веб-сервер и браузер. Рассмотрены непосредственно принципы балансировки, их плюсы и минусы, описана их реализация и применимость к различным сервисам. Исследованы алгоритмы балансировки нагрузки для методов DNS Round Robin, Direct Routing, Редирект. По результатам реализации модели для кластерной системы получены оценки уменьшения эффекта осцилляции между ее перегруженным и нормальным состояниями. В заключение даны рекомендации по использованию реализованной модели сетевой системы в различных промышленных сервисах обработки пользовательских запросов.
Ключевые слова: балансировка нагрузки, дискретно-событийное моделирование, модельно-ориентированное проектирование, вероятностное моделирование

1 Introduction

In many Internet projects, web server is running on a single physical server gradually limiting performance achieved by the server. The most obvious solution in this case is to upgrade hardware equipment. But if one goes this way, then after some time it becomes impossible to upgrade equipment furhter because of limited space and cost efficiency. Therefore, one needs to convert his project to clustered model instead signle server one. Clustering is beyond the scope of this work and thoroughly described in (Teo, 2001: 185-195).
Now the question arises is how to balance load between servers. In addition to direct load distribution is necessary to solve a number of issues. These include increasing resiliency (uninterrupted operation of the project in case of failure of one of servers), and protection against certain types of attacks. For example, from the opening large number of "empty"connections, where nothing is transmitted. In given work considered three algorithms implementing load balancing methods and they modeled using technology of Model-Based Design.
Working on the basis of the model is becoming more and more popular trend in development of a software for systems with clustered resources. On websites of manufacturers of software tools, one can find a lot of success stories, reports on improving the efficiency up to 50%, a significant reduction of errors in the design and a more rapid increase in the level of maturity of developed functions just on the basis of development models (Navarre, 2002: 205-216).
Constructed models allow to model three methods of load balancing in distributed systems, namely DNS Round Robin, Direct Routing, redirect. Of course all three models can be implemented in a cluster environment.

2 Related works

All of reviewed methods and works use queuing networks and their respective algorithms based on the queuing theory (Mitrani, 2004). Studied algorithms of load balancing can be classified into three categories: static, dynamic and adaptive (Al-Amri, 2002: 165-178).
A system described in work (Barros, 2007: 241-255) is a dynamic method which executes programs solving the problem based on "divide-and-conquer"approach on systems with distributed memory. Given system provides primitives for organization of parallel operation of program and uses Random Stealing algorithm (Wrzesinska, 2007: 425-436) for distribution of load between clusters.

Random stealing algorithm fully distributed load balancing algorithm which initiates connection as soon as task being finished and load balancing partner will be chosen randomly among neighbor nodes. Proof of stability of the algorithm for systems with full graph structure written in (Wimmer, 2013: 315).
Another system described in (Eckstein, 2015: 429-469) built as a framework written in $C{+}{+}/MPI$ in order to implement general algorithms of branch and bound method (Krislock, 2017: 1-23). It uses Inversion of Control principle (Martin, 2014) for interaction with code of branch and bound method. Given system optimized in a way that not all nodes are participate in load balancing, only nodes load of which deviates from average on a predefined value. Information about load of nodes distributed through complex mechanism based on linked tree between nodes.
Paper (Aversa, 2005: 1034-1047) offers an algorithm of load distribution for master-slave systems and possesses a hierarchical structure. System consists of three types of nodes: workers, masters and controller. Master defines idle node and sends a task to it from his local pool of subtasks. Worker receives this subtask and executes it sequentially and after finishing sends back results. Job of controller is to monitor load of masters which summarizes load of their respective workers and retranslates subtasks between master nodes if necessary.
Work (Mazzucco et al., 2007) studies issue of building optimal tree structure for distributed networks of grid-services. A system of many nodes considered using queuing networks theory, an average processing time of a single job calculated in an open model for different configurations of interactions between nodes. The structure of systems represented as a tree for organization of interaction between nodes using master-slave paradigm.
Paper (Kameda, 1997: 35-97) arises response time minimization problem for distributed system. System modeled as queuing network consisting of $M/M/1$ servers merged into arbitrary topology network. In paper presented algorithm for parametric by t study of optimal distribution of jobs on servers with complexity $O(n^2M)$ and algorithm of construction of optimal distribution for fixed t of $O(mMlogn)$ complexity where M is a complexity of finding optimal solution (Kontogiannis, 2014: 144).
Mean value analysis method (Geist, 1982: 67) applied to system with closed queuing networks for calculation of parameters. There are exact and approximate methods of mean value analysis. Last reviewed paper considers exact method for closed networks with single class of requests.

3 Material and methods

There are quite a bunch of different algorithms and methods of load balancing. Selecting specific algorithm must be justified, first, by specifics of certain project, second, arisen from goals of the project. In order to find theese goals it necessary to focus on following (Waraich, 2008: 1263-1265) when considering appropriate method for one's own project:

- Justness. It must be guaranteed that for processing of each request system resources allocated and do not allow situation when single request being processed and others wait for their turn;

- Effectiveness. All servers processing requests must be busy 100%. It is desirable not to allow situation when one of servers being idle waiting requests, despite given situation is quite common and this property cannot be always achievable.

- Minimizing request execution time. Time between start of processing of single request (its queuing for processing) and finishing of it must be minimized.

- Minimizing response time. Time of response to user request also must be minimized.

Other desirable (but not necessary) properties of an algorithm are following:

- Predictability. It is necessary to clearly understand in which situations and at which loads algorithm will be efficient for solving given tasks;

- Even distribution of system resources;

- Scalability. Algorithm must preserve its state and stay operational at sudden increase of incoming load.

3.1 DNS Round Robin

The easiest method of balancing is the use of DNS Round Robin algorithm (Borkar, 2011: 198-201). The essence of it is that it creates multiple DNS-records of type A for the domain in the DNS-server. DNS-server returns the record of type A in an alternating cyclic order (Figure 1).

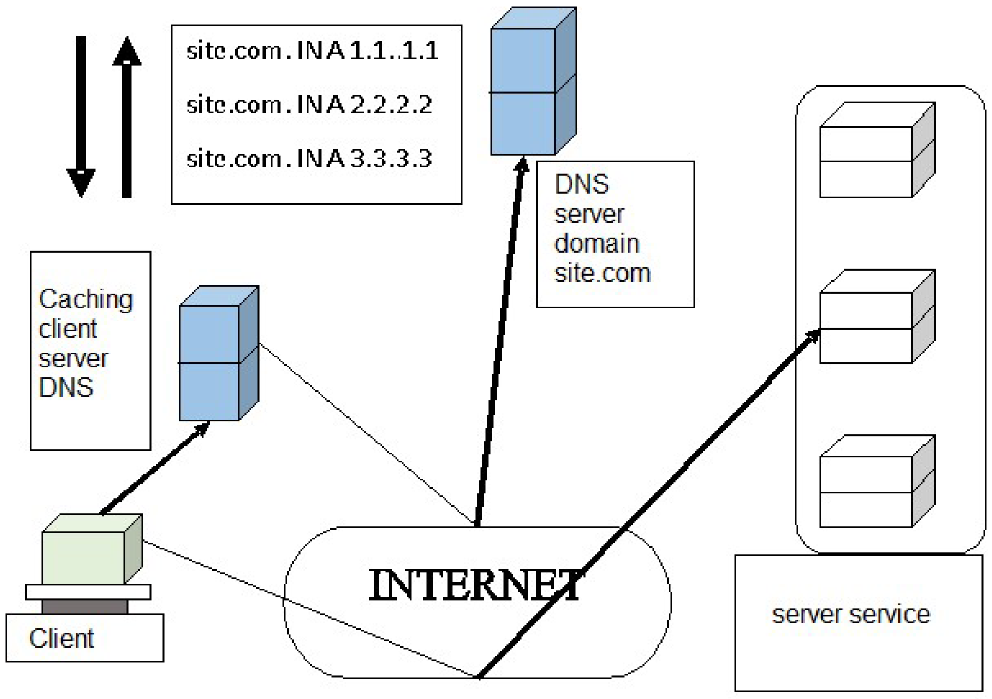


Figure 1 – Load balancing with DNS Round Robin

Method Pros

- It absolutely does not depend on the high-level protocol. That is, this method can be used by any protocol, where an appeal to the server is named;
- Method does not depend on the server load. Due to the fact that there is a DNS-caching server, we do not care how much we will have customers – at least one or millions;
- Method does not require communication between servers. Therefore, it can be used for balancing the local (this balancing of servers within a data center, for example) and for the global balancing, when we have a few data centers, where server together almost have nothing to do;
- The main advantage of the method - it is a low cost solution. If we have a project, domain, DNS-server, we only need to add some records in DNS, to move to this method of balancing.

Method Cons

- It is difficult to shut down servers that do not respond or have failed. VDNS there caching. Recording removed, and customers no longer use it only after a time, which is given by the parameter TTL (Time To Live) in the DNS-zone. In addition, some providers have the DNS-servers that cache entry is forced to a much longer time. We even faced with a situation where entry has been removed from the DNS, and on her customers continued to go another year;
- It is very difficult to distribute the load between servers in the correct proportions. The only way to do this is to provide for each server on several IP-addresses so that their number was in proportion to that part of the load, which should go to them. This is a minus, as the IP-address, we usually do not have much.

3.2 Direct Routing

On balancer having a certain IP-address and responds to the ARP, comes first packet connection. It determines that he was first. Need an algorithm it is sent to the correct server, changing the MAC-address of the place of destination (destination address). Then, the IP-address is written in a certain connection table. If this is not the first packet, it is simply search the connection table. It turns any server processes this compound, and the package is recovering there. The most common solution is now software implementations of this method is called the Linux Virtual Server. The URL-terminology, this method of balancing is called direct routing (Direct Routing, Figure 2).

Method Pros

- Independence from the high-level protocol. One can balance using HTTP, FTP or SMTP – the difference will be negligible;
- There balancing method without a dedicated load balancer. With a small number of servers may be relevant;

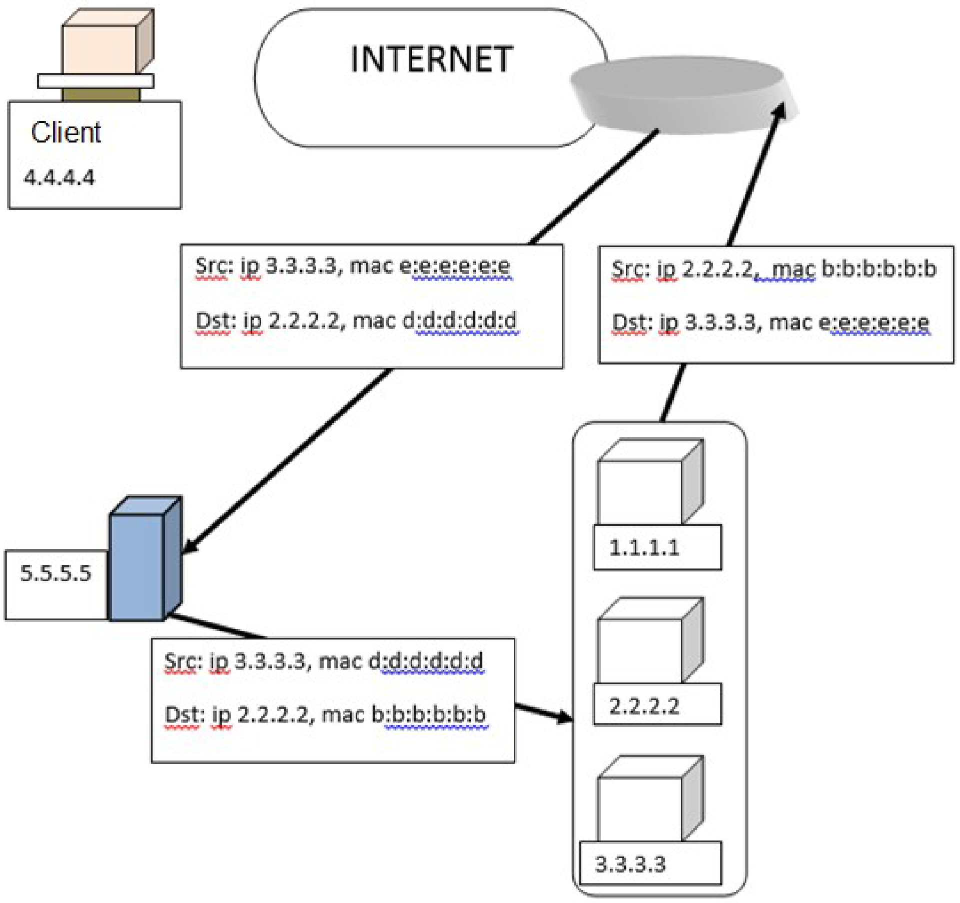

INTERNET
Client
4.4.4.4
Src: ip 3.3.3.3, mac e:e:e:e:e:e
Dst: ip 2.2.2.2, mac d:d:d:d:d:d
Src: ip 2.2.2.2, mac b:b:b:b:b:b
Dst: ip 3.3.3.3, mac e:e:e:e:e:e
5.5.5.5
1.1.1.1
Src: ip 3.3.3.3, mac d:d:d:d:d:d
Dst: ip 2.2.2.2, mac b:b:b:b:b:b
2.2.2.2
3.3.3.3

Figure 2 – Load balancing with Direct Routing

- It is possible to send the replies by balancer. Given that, for example, the protocol HTTP response size is typically an order of magnitude larger than the size of the request, then there is a fairly strong economy of resources;

- Relatively low resource consumption.

Method Cons
The obvious disadvantage of this method is that all servers must be in the water and the same network segment. Need a specific configuration of servers and network equipment. Therefore, this method is not always convenient and is applicable.

3.3 Redirect

There is some balancer that when referring to the service (eg, http://site.com) gives the client a redirect to a particular server (for example, http://server2.site.com). In the case of HTTP it will look like "HTTP redirect 302". Thus, the redirect code will look like "temporarily moved"(moved temporary, Figure 3).

Method Pros

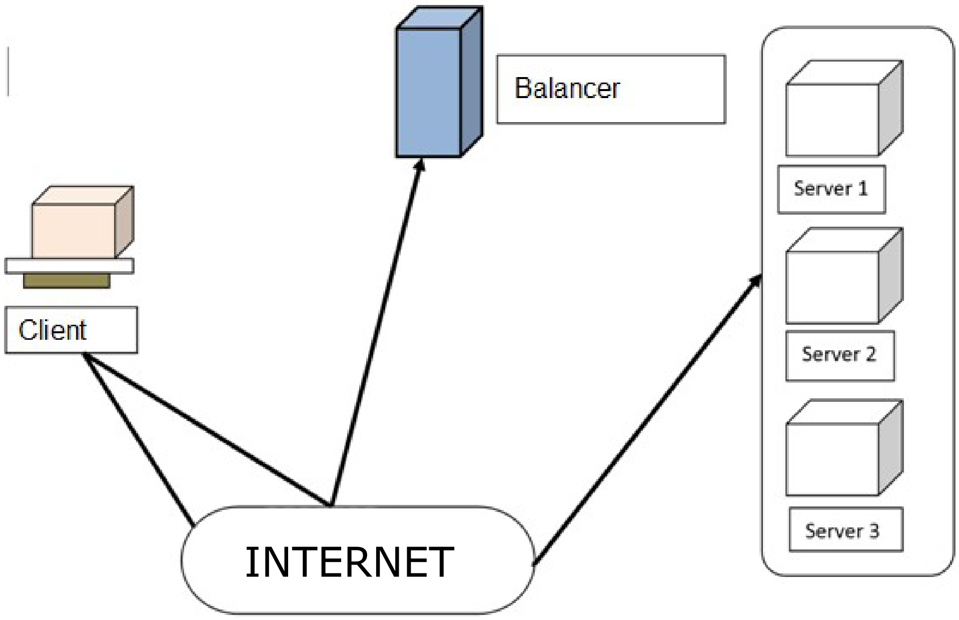


Figure 3 – Load balancing by redirecting

- If the request is sufficiently "heavy it sometimes makes sense to use a redirection, even for global balance. We have a load balancer that sends via redirects requests for processing in different data centers;
- The method also allows users to distribute different types of requests to different servers;
- Queries may well be analyzed.

Method Cons

- It is, as has been said, applies to a very small number of high-level protocols;
- For the client to each request, it turns out, it made two requests;
- One-to our redirector, one- to the server that handles the connection. This increases the time in which the client will receive a final answer to customers request.

4 Results and discussion

Analytical representation of algorithms implemented in the form of model-oriented design models with intention to further verify them. For this purpose, the model-oriented design tool Mathworks Simulink (Simulink, 2016) was used. For given paper currently only DNS Round Robin algorithm implemented in model's Scheduler block, but other algorithm implementations also quite trivial to build and will be implemented further.
The SimEvents framework (SimEvents, 2016) in Simulink is designed for visual modeling of processes with discrete-event semantics, including simulating the behavior of a multiprocessor / multicore system with unstable resources. This framework allows to simulate the effects

of planning (delays, etc.) and to explore the design area of a running task by analyzing the impact of decisions made in planning on the performance of management design. The framework provides control algorithms and makes it possible to smoothly transition to implementing the algorithm from the model through automatic code generation technologies.

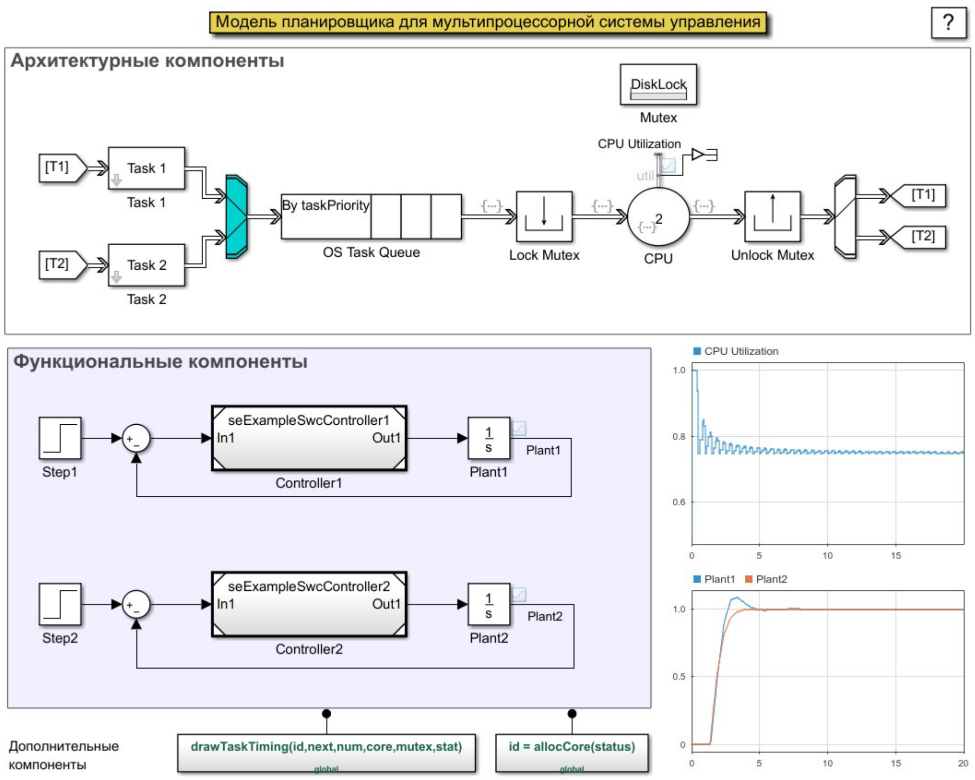
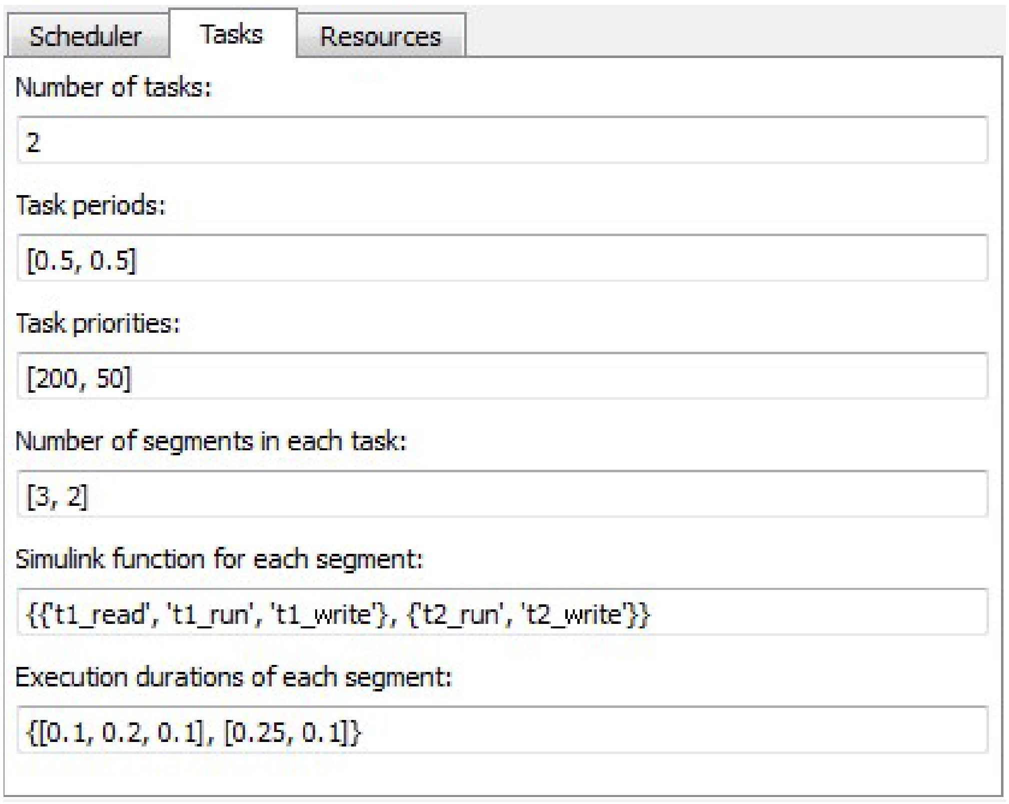


Figure 4 – Scheduler implementation of multicore control system with Simulink model

Figure 4 shows the Simulink model using the SimEvents framework containing a block of architectural components and a block of functional components. The scheduler implicitly implies the existence of a Task definition, which is a real-time task with the following attributes: identifier, period, priority, segments (subtasks). The scheduler simulates a homogeneous multiprocessor system and is defined by the following properties: the number of processes, the planning policy, the mutually exclusive resources. The implementation of the architectural block of the scheduler is built on a high-level discrete-event language supported by Simulink. In turn, this language relies on the MATLAB interpreter, which allows to implement own scheduling policies for processors, which was done, i.e. The implementation of the standard planning policy has been replaced. In our case, these were the implementations of the 3 algorithms described above in MATLAB.
The SimEvents Scheduler permits clients to determine errands as some parameters additionally as appeared in Figure 5.

These parameters characterize a homogeneous multicore framework with two centers. The scheduler is setup to play out a need based approach. The framework has two control undertakings, with:



Number of tasks:
2

Task periods:
[0.5, 0.5]

Task priorities:
[200, 50]

Number of segments in each task:
[3, 2]

Simulink function for each segment:
{{'t1_read', 't1_run', 't1_write'}, {'t2_run', 't2_write'}}

Execution durations of each segment:
{[0.1, 0.2, 0.1], [0.25, 0.1]}

Figure 5 – Parameters used in Scheduler block for DNS Round Robin method

- Task 1 for Controller1 is arranged with a time of 0.5 second, need of 200 and 3 sections, whose capacities and execution lengths are (t1_read, t1_run, t1_write) and (0.1, 0.2, 0.1) second individually;
- Task 2 for Controller2 is arranged with a time of 0.5 second, need of 50, 2 sections, whose capacities also, execution lengths are (t2_run, t2_write) and (0.25, 0.1) second individually. The Scheduler is made as a MATLAB Discrete-Event System. It executes the multi-center scheduler as takes after:
- An assignment occurrence is demonstrated as an element that is intermittently made, executed, and demolished toward the end of execution. Properties and runtime conditions of an assignment occasion are put away as information of the element;
- The assignment line of the working framework and the multicore processor itself are demonstrated as two element stockpiles;
- Task occurrences (elements) are made occasionally inside the assignment line. They are put away in a requested grouping as required by the planning approach;
- The booking calculation is acknowledged by planning occasions on the undertaking substances. Occasions cause errands (elements) that are prepared to execute to be sent from the assignment line to the processor (the second element stockpiling);

- An executing assignment remains in the processor for the time period as determined by the execution length parameter. Such execution term is reproduced by utilizing a clock occasion. At the point when the clock finishes, its occasion activity executes the comparing assignment work (Simulink Function).

The Scheduler permits appointing a subjective number of centers and investigating how that effects SimEvents framework execution. In the primary situation two centers have been allocated to execute the two control errands. Fig. **??** shows parameters of the Scheduler obstruct for this setup.
With adequate preparing limit, both shut circle control frameworks perform acceptably when the set point is changed from 0 to 1. Fig. **??** incorporates a reaction outline (on the left-hand side) and a timing outline (on the right-hand side):

- The reaction outline demonstrates the reaction of the two controllers with Controller1 in the top diagram, and Controller2 in the base diagram;

- The planning graph demonstrates usage of centers and assets. A hued bar shows execution of a fragment of errand. The position of a bar on the flat pivot shows time of begin and consummation of the fragment. The position of the bar on the vertical pivot shows standardized errand consummation prior and then afterward executing the fragment.

The planning graph of Figure 6 shows that errands are prepared simultaneously with centers having medium and adjusted usages. See that Task2 allocated to Core1 since it has higher need and, along these lines, allocated before Task1.

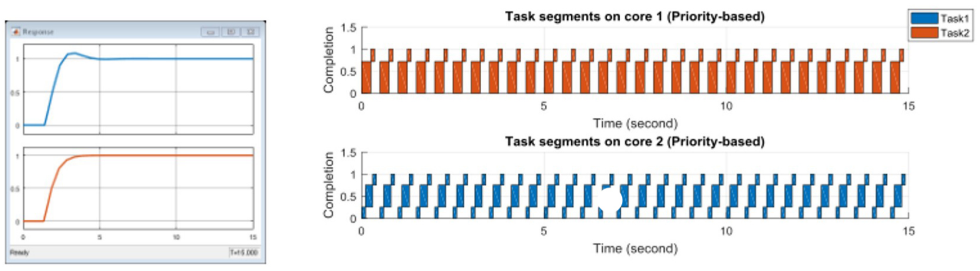


Figure 6 – Task scheduler output and scheduler performance (2 cores) with DNS Round Robin scheduling

5 Conclusion

Therefore, common software-balancing algorithms have been considered, which are often used to solve the problem of load balancing servers, websites etc. In traditional web server architectures DNS balancer distributes requests to the server based on the status of their workload. Because the Web servers must inform the DNS server about the status of their load from time to time, the so-called buffer download busy often to reduce the frequency of updates. Without proper attention, excessive use of loading buffer may lead to excessive oscillation within the cluster. All three discussed the method helps reduce the oscillation effect on Web servers that has been tested through the implementation of their models using Model-Based Design.

References

[1] *Al-Amri, M. S., and S. E. Ahmed.* "New job selection and location policies for load-distributing algorithms."International Journal of Network Management 12, no. 3 (2002): 165-78. doi:10.1002/nem.428.

[2] *Aversa, R., B. Di Martino, N. Mazzocca, and S. Venticinque.* "A hierarchical distributed-shared memory parallel Branch&Bound application with PVM and OpenMP for multiprocessor clusters."Parallel Computing 31, no. 10-12 (2005): 1034-047. doi:10.1016/j.parco.2005.03.010.

[3] *Borkar, G. M., M. A. Pund, and P. Jawade.* "Implementation of round robin policy in DNS for thresholding of distributed web server system."Proceedings of the International Conference & Workshop on Emerging Trends in Technology – ICWET '11, 2011, 198-201. doi:10.1145/1980022.1980067.

[4] *Barros, F. J.* "Modeling and simulation of parallel adaptive divide-and-conquer algorithms."The Journal of Supercomputing 43, no. 3 (2007): 241-55. doi:10.1007/s11227-007-0143-3.

[5] *Eckstein, J., W. E. Hart, and C. A. Phillips.* PEBBL: an object-oriented framework for scalable parallel branch and bound."Mathematical Programming Computation 7, no. 4 (2015): 429-69. doi:10.1007/s12532-015-0087-1.

[6] *Geist, R., and K. Trivedi.* "Queueing Network Models in Computer System Design."Mathematics Magazine 55, no. 2 (1982): 67. doi:10.2307/2690049.

[7] *Kameda, H., J. Li, C. Kim, and Y. Zhang.* "Overall Optimal Load Balancing vs. Individually Optimal Load Balancing."Optimal Load Balancing in Distributed Computer Systems Telecommunication Networks and Computer Systems, 1997, 35-97. doi:10.1007/978-1-4471-0969-3_2.

[8] *Kontogiannis, S., and A. Karakos.* "ALBL: an adaptive load balancing algorithm for distributed web systems."International Journal of Communication Networks and Distributed Systems 13, no. 2 (2014): 144. doi:10.1504/ijcnds.2014.064041.

[9] *Krislock, N., J. Malick, and F. Roupin.* "BiqCrunch."ACM Transactions on Mathematical Software 43, no. 4 (2017): 1-23. doi:10.1145/3005345.

[10] *Martin, R. C.* Agile software development principles, patterns, and practices. Harlow: Pearson Education Ltd, 2014.

[11] *MathWorks* "SimEvents User's Guide."The MathWorks Inc., Natick, MA, USA (2016): 208 p.

[12] *MathWorks* "Simulink User's Guide."The MathWorks Inc., Natick, MA, USA (2016): 3290 p.

[13] *Mazzucco, M., I. Mitrani, J. Palmer, M. Fisher, and P. Mckee.* "Web Service Hosting and Revenue Maximization."Fifth European Conference on Web Services (ECOWS'07), 2007. doi:10.1109/ecows.2007.8.

[14] *Mitrani, I.* Probabilistic modelling. Cambridge: Cambridge University Press, 2004.

[15] *Navarre, D., P. Palanque, and R. Bastide.* "Model-Based Interactive Prototyping of Highly Interactive Applications."Computer-Aided Design of User Interfaces III (2002): 205-16. doi:10.1007/978-94-010-0421-3_18.

[16] *Teo, Y. M., and R. Ayani.* "Comparison of Load Balancing Strategies on Cluster-based Web Servers."Simulation 77, no. 5-6 (2001): 185-95. doi:10.1177/003754970107700504.

[17] *Waraich, S. S.* "Classification of Dynamic Load Balancing Strategies in a Network of Workstations."Fifth International Conference on Information Technology: New Generations (itng 2008), 2008, 1263-265. doi:10.1109/itng.2008.166.

[18] *Wimmer, M., D. Cederman, J. Larsson TrГ¤ff, and P. Tsigas.* "Work-stealing with configurable scheduling strategies."ACM SIGPLAN Notices 48, no. 8 (2013): 315. doi:10.1145/2517327.2442562.

[19] *Wrzesinska, G., A. Oprescu, T. Kielmann, and H. Bal.* "Persistent Fault-Tolerance for Divide-and-Conquer Applications on the Grid."Euro-Par 2007 Parallel Processing Lecture Notes in Computer Science: 425-36. doi:10.1007/978-3-540-74466-5_46.