RspSim – A Simulation Model of the Reliable Server Pooling Framework

Thomas Dreibholz, Martin Becke, Hakim Adhari, Erwin P. Rathgeb
University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstraße 29, 45326 Essen, Germany
{dreibh, martin.becke, hakim.adhari, rathgeb}@iem.uni-due.de

***Abstract*—This code contribution paper provides an overview of the RSPSIM model, which is a simulation model for the Reliable Server Pooling (RSerPool) framework. RSerPool denotes an IETF standard for the management of server pools and sessions with these pools. Such mechanisms are also crucial in the context of cloud computing research.[1][2]**

Keywords: Reliable Server Pooling, Simulation, Evaluation, Model, Cloud Computing

I. Introduction

Service availability is becoming increasingly important in the Internet. However, there had been no generic, standardised approaches for managing the availability of Internet-based services. Instead, each developer of an availability-critical application had to continuously re-invent the wheel again and again. To overcome this problem, the IETF RSerPool Working Group had been founded to develop Reliable Server Pooling (RSerPool, [1], [2]), an application-independent framework for managing server pools and sessions.

Clearly, in order to evaluate RSerPool, an OMNET++-based simulation model – called RSPSIM – had been developed. It has already been used for a couple of research publications, e.g. [2]–[7]. In this code contribution, this model is released to the public. Particularly, it may be useful in the context of cloud computing – which has to solve very similar problems of server pool and session management.

II. The RSerPool Architecture

An overview of the RSerPool architecture [1], [8] with its three types of components is depicted in Figure 1: a server in a pool is called *Pool Element* (PE), a client is denoted as a *Pool User* (PU). The *Handlespace* – which is the set of all pools – is managed by redundant *Pool Registrars* (PR). Within the handlespace, each pool is identified by a unique *Pool Handle* (PH).

A. Registrar Operations

PRs of an *Operation Scope* synchronise their view of the handlespace by using the Endpoint haNdlespace Redundancy Protocol (ENRP) [9]. An operation scope is restricted to a single administrative domain. That is, all of its components are under the control of the same authority (e.g. a company). This property leads to small management overhead [10], which also allows for RSerPool usage on devices with limited memory and CPU resources (e.g. telecommunications equipment). Nevertheless, PEs may be distributed globally to continue their service even in case of localised disasters [11] (e.g. an earthquake).

[1]Parts of this work have been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG).
[2]The authors would like to thank Xing Zhou for her friendly support.

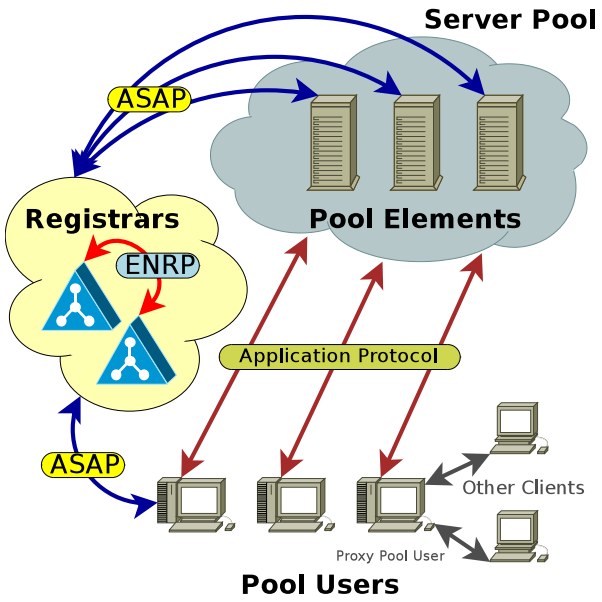

Figure 1. The RSerPool Architecture

B. Pool Element Operations

PEs choose an arbitrary PR of the operation scope to register into a pool by using the Aggregate Server Access Protocol (ASAP) [12]. Within its pool, a PE is characterised by its PE ID, which is a randomly chosen 32-bit number. Upon registration at a PR, the chosen PR becomes the Home-PR (PR-H) of the newly registered PE. A PR-H is responsible for monitoring its PEs' availability by keep-alive messages (to be acknowledged by the PE within a given timeout) and propagates the information about its PEs to the other PRs of the operation scope via ENRP updates. PEs re-register regularly (in an interval denoted as *Registration Lifetime*) and for information updates.

C. Pool User Operations

In order to access the service of a pool given by its PH, a PU requests a PE selection from an arbitrary PR of the operation scope, again by using ASAP. The PR selects the requested list of PE identities by applying a pool-specific selection rule, denoted as *Pool Policy*. Two classes of load distribution policies are supported: non-adaptive and adaptive strategies [7], [10], [11], [13], [14]. While adaptive strategies base their selections on the current PE state (which requires up-to-date information), non-adaptive algorithms do not need such data.

III. The RspSim Model

A. The Modules

The core module of the RSPSIM simulation model is the *TransportNode* module. It contains a lightweight reliable message forwarding service. A future version could simply replace

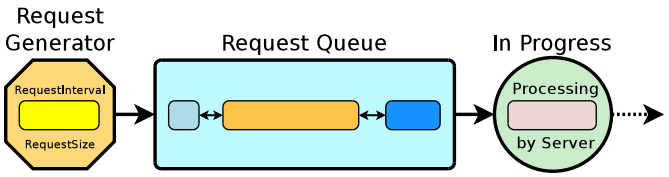

Figure 2. The CalcApp Pool User Operation

this service by using e.g. SCTP/IP of the INET FRAMEWORK. Based on *TransportNode*, modules for PRs, PEs and PUs have been realised as compound modules. Each of them contains a sub-module of *TransportNode*, a sub-module for the RSerPool protocol(s) and a sub-module for the application service.

B. The CalcApp Service

The currently only application service has been denoted as CalcApp ("calculation application"). Each PE has a request handling *Capacity*, given in the abstract unit of calculations per second. Each request consumes a certain number of calculations, denoted as *Request Size*. A PE can handle multiple requests simultaneously, in a processor sharing mode as provided by multitasking operating systems. Each PU generates a new request in an interval denoted as *Request Interval*. Requests are queued and sequentially assigned, as illustrated in Figure 2. The total delay for handling a request d_{Handling} is defined as the sum of queuing delay d_{Queuing}, startup delay d_{Startup} (dequeuing until reception of acceptance acknowledgement) and processing time $d_{\mathrm{Processing}}$ (acceptance until finish):

$$d_{\mathrm{Handling}} = d_{\mathrm{Queuing}} + d_{\mathrm{Startup}} + d_{\mathrm{Processing}}.$$

That is, d_{Handling} not only incorporates the time required for processing the request, but also the latencies of queuing, server selection and message transport. The user-side performance metric is the *Handling Speed*, which is defined as:

$$\mathrm{HandlingSpeed} = \frac{\mathrm{RequestSize}}{d_{\mathrm{Handling}}}.$$

The number of PUs can be given by the ratio between PUs and PEs (*PU:PE Ratio* puToPERatio), which defines the request handling parallelism. The average *System Utilisation U* (for NumPEs servers and total pool capacity PoolCapacity) can be calculated as:

$$\mathrm{U} = \mathrm{NumPEs} * \mathrm{puToPERatio} * \frac{\frac{\mathrm{RequestSize}}{\mathrm{RequestInterval}}}{\mathrm{PoolCapacity}}.$$

Obviously, the primary provider-side performance metric is the system utilisation, since only utilised servers gain revenue. In practise, a well-designed client/server system is dimensioned for a certain *Target System Utilisation* of e.g. 50%.

C. Simulations with SIMPROCTC

For easier simulation parametrisation, run distribution and results analysis, the model sources also provide scripts to utilise the Simulation Processing Tool-Chain SIMPROCTC [15] for this task. A particularly interesting fact is that SIMPROCTC uses the real RSerPool implementation RSPLIB [8] to distribute the simulation runs in a compute pool.

IV. THE SOURCE PACKAGE

The RSPSIM package `rspsim-<version>.tar.gz`, which is available from the project website [16], consists of:

- The full sources of the RSPSIM model based on OMNET++ 4.2 (in the directory `model`),
- A simple example (`test1.ini`) and
- A SIMPROCTC-based example (in the directory `toolchain`).

The included file `README` describes how to compile the model and run the examples. Some more details on the implementation can be found in [8].

V. CONCLUSIONS

The Reliable Server Pooling (RSerPool) framework is the new IETF standard for server pool and session management. Particularly, the tasks provided by RSerPool are also useful in the context of cloud computing research. The RSPSIM simulation model provides a simulation environment for RSerPool systems. It is released to the OMNET++ community by this code contribution.

REFERENCES

[1] P. Lei, L. Ong, M. Tüxen, and T. Dreibholz, "An Overview of Reliable Server Pooling Protocols," IETF, Informational RFC 5351, Sept. 2008, ISSN 2070-1721.

[2] T. Dreibholz and E. P. Rathgeb, "Overview and Evaluation of the Server Redundancy and Session Failover Mechanisms in the Reliable Server Pooling Framework," *International Journal on Advances in Internet Technology (IJAIT)*, vol. 2, no. 1, pp. 1–14, June 2009, ISSN 1942-2652.

[3] T. Dreibholz, X. Zhou, M. Becke, J. Pulinthanath, E. P. Rathgeb, and W. Du, "On the Security of Reliable Server Pooling Systems," *International Journal on Intelligent Information and Database Systems (IJIIDS)*, vol. 4, no. 6, pp. 552–578, Dec. 2010, ISSN 1751-5858.

[4] X. Zhou, T. Dreibholz, F. Fa, W. Du, and E. P. Rathgeb, "Evaluation and Optimization of the Registrar Redundancy Handling in Reliable Server Pooling Systems," in *Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA)*, Bradford/United Kingdom, May 2009, pp. 256–262, ISBN 978-0-7695-3638-5.

[5] T. Dreibholz, E. P. Rathgeb, and X. Zhou, "On Robustness and Countermeasures of Reliable Server Pooling Systems against Denial of Service Attacks," in *Proceedings of the 7th International IFIP Networking Conference*, ser. Lecture Notes in Computer Science, vol. 4982. Singapore: Springer, May 2008, pp. 586–598, ISBN 978-3-540-79548-3.

[6] P. Schöttle, T. Dreibholz, and E. P. Rathgeb, "On the Application of Anomaly Detection in Reliable Server Pooling Systems for Improved Robustness against Denial of Service Attacks," in *Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN)*, Montréal, Québec/Canada, Oct. 2008, pp. 207–214, ISBN 978-1-4244-2413-9.

[7] T. Dreibholz and E. P. Rathgeb, "On the Performance of Reliable Server Pooling Systems," in *Proceedings of the IEEE Conference on Local Computer Networks (LCN) 30th Anniversary*, Sydney/Australia, Nov. 2005, pp. 200–208, ISBN 0-7695-2421-4.

[8] T. Dreibholz, "Reliable Server Pooling – Evaluation, Optimization and Extension of a Novel IETF Architecture," Ph.D. dissertation, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, Mar. 2007.

[9] Q. Xie, R. R. Stewart, M. Stillman, M. Tüxen, and A. J. Silverton, "Endpoint Handlespace Redundancy Protocol (ENRP)," IETF, RFC 5353, Sept. 2008, ISSN 2070-1721.

[10] T. Dreibholz and E. P. Rathgeb, "An Evaluation of the Pool Maintenance Overhead in Reliable Server Pooling Systems," *SERSC International Journal on Hybrid Information Technology (IJHIT)*, vol. 1, no. 2, pp. 17–32, Apr. 2008, ISSN 1738-9968.

[11] ——, "On Improving the Performance of Reliable Server Pooling Systems for Distance-Sensitive Distributed Applications," in *Proceedings of the 15. ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, ser. Informatik aktuell. Bern/Switzerland: Springer, Feb. 2007, pp. 39–50, ISBN 978-3-540-69962-0.

[12] R. R. Stewart, Q. Xie, M. Stillman, and M. Tüxen, "Aggregate Server Access Protcol (ASAP)," IETF, RFC 5352, Sept. 2008, ISSN 2070-1721.

[13] T. Dreibholz and M. Tüxen, "Reliable Server Pooling Policies," IETF, RFC 5356, Sept. 2008, ISSN 2070-1721.

[14] T. Dreibholz and E. P. Rathgeb, "The Performance of Reliable Server Pooling Systems in Different Server Capacity Scenarios," in *Proceedings of the IEEE TENCON*, Melbourne/Australia, Nov. 2005, ISBN 0-7803-9312-0.

[15] T. Dreibholz, X. Zhou, and E. P. Rathgeb, "SimProcTC – The Design and Realization of a Powerful Tool-Chain for OMNeT++ Simulations," in *Proceedings of the 2nd ACM/ICST International Workshop on OMNeT++*, Rome/Italy, Mar. 2009, pp. 1–8, ISBN 978-963-9799-45-5.

[16] T. Dreibholz, "Thomas Dreibholz's RSerPool Page," 2012. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/rserpool/