2023 IEEE International Conference on Software Services Engineering (SSE) | 979-8-3503-4075-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/SSE60056.2023.00040

Component-Based Message Passing for Java: Developments, Achievements, and Impact

Vladimir Getov
Distributed and Intelligent Systems Research Group
School of Computer Science and Engineering
University of Westminster, London, United Kingdom
V.S.Getov@westminster.ac.uk

***Abstract*—The pioneering research on message passing for Java (MPJ) which started after 1995 provided a crucially important framework and programming environment for parallel and distributed computing with Java. This framework resulted in an industry standard specification and a novel MPJ-based hierarchical development methodology for a new generation of large-scale distributed systems. The invention of a novel component-based model and methodology for rapid distributed software development and execution based on the MPJ work and achievements are the core contributions presented in this paper. Based on the high-performance Java component-based model, concepts and research results, grid, cloud, and extreme-scale computing represent a fundamental shift in the delivery of information technology services that has permanently changed the computing landscape.**

***Keywords*—message passing for Java, software components, cloud and grid computing, economic and social impact**

I. Introduction

Released in 1995, the Java programming language was rapidly adopted by industry and end users because of its portability and internet programming support. However, Java did not provide support for message passing. This capability is vitally important for parallel and distributed memory computing and its absence substantially reduced Java's potential [1]. By contrast, efficient message passing support had already been captured in the MPI (message-passing interface) standard for other programming languages such as C, C++, and Fortran.

To alleviate this difficulty, various early projects including our own work started the development of Java message-passing systems. Then, a single MPI-like API specification and reference implementation were developed by the Message Passing for Java (MPJ) International Working Group as part of the JavaGrande Forum [7]. This group also developed a methodology for building mixed-language MPJ applications which evolved from three approaches: (a) wrapping of existing MPI libraries via hand-written software [5]; (b) automatic code generation of the wrapper interface by a novel tool-generator [6]; and (c) development from scratch of the MPJ libraries in Java [4]. The development of all three approaches implemented the MPJ specification which successfully resembled MPI, providing symmetric message passing for Java [2].

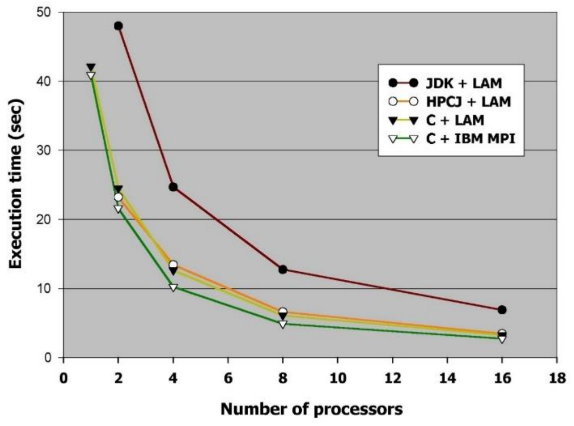

Fig. 1. Ping-pong timings for different MPI environment on IBM SP2.

However, as shown in Fig. 1 and Fig. 2, due to the lower performance of the Java platforms at that time, the adoption of MPJ took longer and had to wait for the development and introduction of other relevant areas such as grid, cloud, and extreme-scale computing [3, 5].

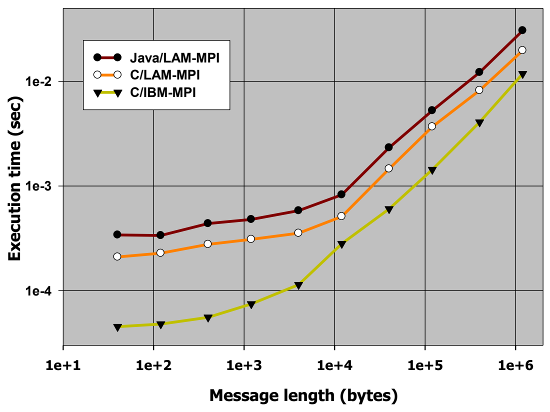

Fig. 2. NPB IS kernel with JDK, HPCJ, and MPI on IBM SP2.

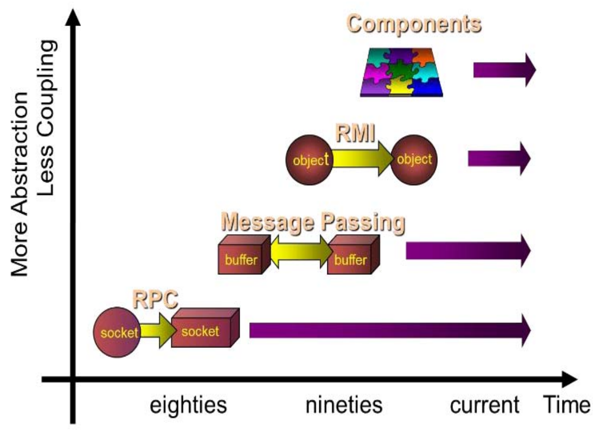

Fig. 3. Roadmap of communication frameworks.

II. Further Developments

The MPJ International Working Group was then invited to join the Global Grid Forum and continue further this research by expanding the application of MPJ into modern large-scale distributed systems such as grids and clouds, both of which allow for the analysis of huge data sets. Tackling the scalability and productivity challenges, the team formally specified software components as novel building blocks. As shown in Fig. 3 this led to the introduction of component-based approaches and frameworks that enabled MPJ to provide the interconnection mechanisms in complex grid and cloud systems [14].

Continuing work in this area, the next main research challenge was to replace the Common Component Architecture (CCA) model [15]. Available since early 2000, the CCA provided only limited support for a single coupling of components in a 2-dimensional space. The initial idea was to develop a novel abstract approach for hierarchical (multiple) composition of components in a 3-dimensional space which led to the introduction of a novel MPJ-based hierarchical components composition development methodology for a new generation of large-scale grid and cloud-based distributed systems. This provided the theoretical background for a recursive and efficient component-based development style. The combination of these research contributions to the European CoreGRID project significantly advanced this new field through the development of the hierarchical Grid Component Model (GCM) specification for various distributed computing systems [10]. The team then followed up the GCM specification with proof-of-concept experiments in a development environment that used the hierarchical components MPJ methodology and provided confidence about the higher efficiency of this novel approach [11].

Building on this work, the Distributed and Intelligent Systems Research Group (DIS RG) was a main partner in the European GridComp project. Working in close collaboration with other partners including INRIA Sophia Antipolis (France), IBM-Research (Switzerland), Atos Origin (Spain), University of Pisa (Italy), and Tsinghua University (China), the GridComp project designed and built a fully functional platform incorporating the ICBE (Integrated Component-Based Environment) prototype for hierarchical components composition [12]. A major contribution of GridComp was the design and implementation of a component-based MPJ framework for rapid development and deployment of efficient grid and cloud computing applications. This work led to four new international standards, approved by ETSI (European Telecommunications Standards Institute): "GCM Interoperability Deployment", Aug 2008; "GCM Interoperability Application Description", Aug 2008; “GCM Fractal ADL”, Mar 2009; “GCM Management API (Java, C, WSDL)”, Mar 2010.

Further results on smart cloud architectures followed the expanded ICBE methodology, which included three approaches for developing GCM applications [13]. The first, a wrapper approach for legacy codes reuse, supports both hand-written and automatically generated code. The second approach componentizes existing applications via appropriate modifications. The third approach is component-based development from scratch. This work confirmed the significantly higher productivity of the component-based MPJ development methodology.

III. Achievements

More recently, MPJ has been the forerunner of the Java binding which has been included since 2014 in the core distribution of the widely used Open MPI library. Open MPI is a software environment which enables the development of very large and complex parallel and distributed applications [8]. Presently, MPJ is also the predecessor of the Java binding in another very popular open-source MPI library – the MVAPICH2 message-passing software environment [9].

IV. Impact

The invention of MPJ resulted in an industry standard specification and a novel component-based hierarchical development methodology which enables the development of very large and complex grid and cloud-based distributed systems. These achievements led to: (a) impact on professional practice and development productivity; (b) significant economic impact; and (c) social impact via the results of the novel component-based application.

A. Professional Practice - Open Source and Standardization

Since August 2014, the MPJ specification has been included in the core distribution of the widely used Open MPI (Open-Source Message Passing Interface) software environment. Developed and maintained by a consortium of research, academic, and industry partners, Open MPI is a suite of open-source software libraries implementing the MPI standard for High Performance Computing (HPC) [16]. The Java binding developed in Open MPI is based on the ground-breaking results of the International MPJ Working Group [2] and incorporates the four ETSI standards that resulted from the GridComp project [11-13]. The Open MPI team have highlighted that these standardized classes of Java are of benefit to an open-source library as they provide a platform-independent way to access host-specific features such as threads, graphics, file management, and networking [8]. In other words, these standardized classes guarantee full compatibility and portability across a wide variety of platforms and applications.

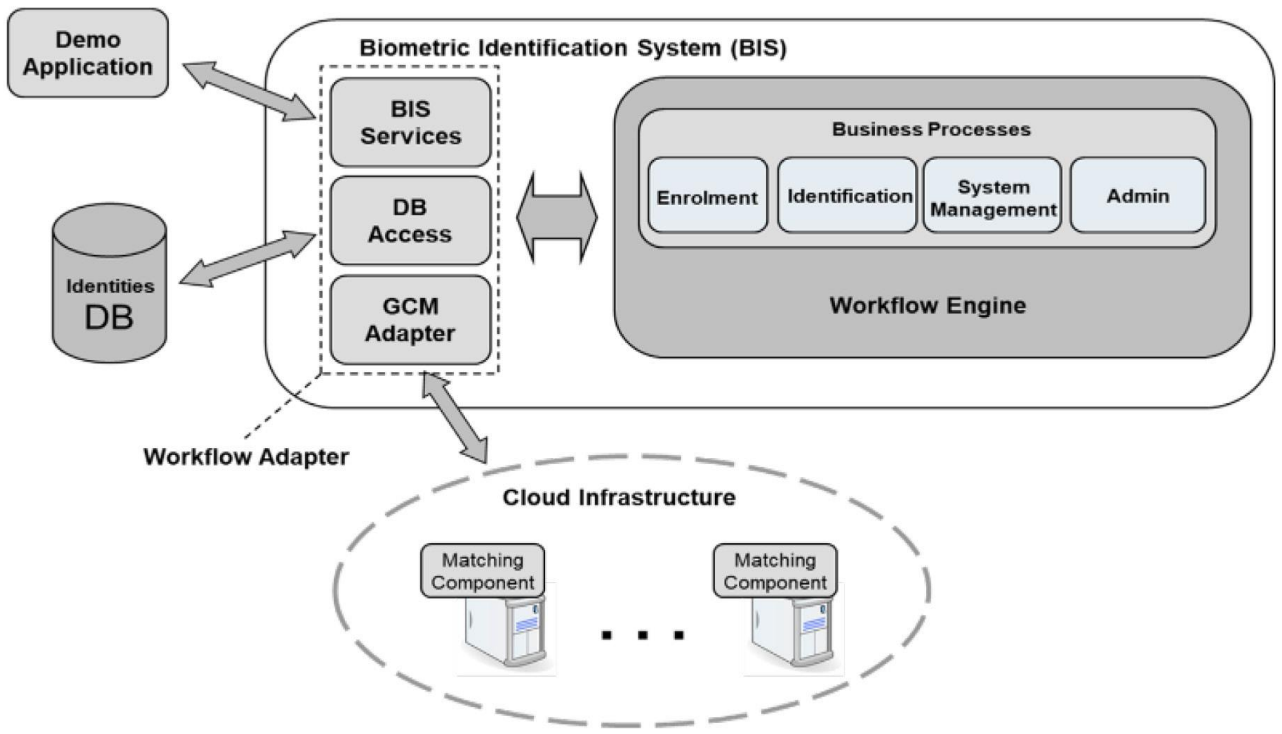

Fig. 4. Security bimetric identification system.

In recent years, Open MPI has become the most popular MPI library, as is evidenced by the MPI International Survey, which received over 800 user responses from 42 different countries between Feb & July 2019. Of the MPI users identified, 85.8% (718 respondents) use Open MPI. As such, through its inclusion in this open-source library, MPJ has been reaching a wide range of users, 80% of which belong to non-profit organizations, including government institutes, as well as software and hardware vendors.

Regarding the significance of this wide range of users, the Open MPI team explain that MPJ was added as it is one of the three key parallel computing technologies upon which HPC depends and has several benefits. For instance, it provides efficient built-in support for threads which increases software development productivity. In addition, some numerical libraries are based on multithreading and MPI itself can benefit from Java because its widespread use makes it likely to find new uses beyond traditional HPC applications. As such, the primary impact on professional practice relates to providing easy access to Java binding for programmers who can exploit these benefits [14].

Secondary impacts arise through specific examples of such usage of MPJ via Open MPI. For instance, the PPF library for parallel particle filtering applications allows application programmers to write quickly and easily shared- and distributed-memory PPF codes in Java, producing the benefit of reducing the cost of traditional particle filters by approximating the likelihood with a mixture of uniform distributions over pre-defined cells or bins.

B. Economic Impact

An example of the significant economic impact on users of the innovative component-based MPJ development methodology is its contribution to the long-term research and development collaboration for IBM's Cloud Computing agenda and strategy, through knowledge exchange activities at IBM Research Centers in Zurich, Almaden, Watson, and Dublin, as well as direct collaborative research. According to the Computer Science Department at IBM's Zurich Research Laboratory s this has resulted in their using and developing further the adopted component-based methodology and development platform which has in turn influenced the professional practice in building complex applications for grid and cloud systems at IBM.

For instance, IBM's Watson supercomputer was designed by reusing the principles developed and standardized by the European Telecommunications Standards Institute as part of the MPJ work [12]. This supercomputer is a unique artificial intelligence system capable of answering questions posed in natural language and its economic significance to IBM is indicated by its global market share of 16.1% placing it amongst the three dominant systems in the machine learning category. IBM Watson has been used by nearly 2,800 companies as of July 2020 due to its usefulness across a variety of application domains. Statistics by revenue show that 39% of Watson users are large companies (>$1000M), 37% are small (<$50M), and 16% are medium-sized, and that they cut across different sectors of industry – the largest being Computer Software (22%), Hospital and Health Care (14%), Higher Education (8%), and Information Technology and Services (7%).

In addition, the Biometric Identification System application (Fig. 4) developed in collaboration with the DIS RG team and delivered by IBM Research – Zurich Laboratory has been reported as having a significant economic impact through its producing quicker return on investment achieved because of the much higher productivity and shorter development cycle provided by the invented component-based methodology and development process.

This biometric identification system contributes to the Global Technology Services segment of IBM's business, which in 2019 comprised the largest share of IBM's total revenue at nearly 36% ($27.4 billion). The use of the component-based MPJ framework provided pre-built template components from which a system can be developed that guaranteed real-time biometric identification functionality over a very large, constantly growing database of enrolled identities (fingerprints). Furthermore, the ICBE [11-13] has been instrumental in continuously monitoring and maintaining the system, which has been a significant aspect of IBM's commercial offering to corporations and governments.

Another company that has economically benefitted from the innovative component-based MPJ methodology, as well as the underpinning ETSI standards, is ActiveEon – a France-based software company that provides innovative open-source solutions for IT automation, acceleration and scalability, big data, distributed computing, and application orchestration. During the GridCOMP project (06/2006 – 02/2009), the partners had initially demonstrated the usefulness of the component-based MPJ development methodology to industry by using it to wrap and Grid-enable aerodynamic wing modelling software at ActiveEon, and to prove the integration of data staging for the input and output files into the sweeping and optimization process for any given configuration.

As stated by ActiveEon, since then the company has been directly exploiting the results of the project, including the GCM and MPJ framework [10], particularly in its ProActive workflows and scheduling open-source middleware which has been used for delivering solutions for a range of commercial customers. Furthermore, the economic impact of the adoption of MPJ's innovations over the years since August 2013, ActiveEon has achieved revenue growth of 18M USD starting with less than 1M USD in 2013 and reaching revenue of 19M USD in 2020. Their number of employees has been increasing significantly over the same period from 30 employees at the end of 2013 to 106 employees in 2020.

The above successful results would not have been possible without the direct contribution to the component-based MPJ development methodology and ETSI standards adopted and used by ActiveEon throughout the 13 years of its existence. Subsequently, the MPJ work has reduced substantially the return of investment cycle of complex distributed applications.

C. Social Impact

IBM specify three key social impacts of their cognitive technology in lengthy case studies featuring real-world usage of the IBM Watson supercomputer that would not exist without the MPJ research results. The areas of interventions listed below provide typical examples:

1) Transforming social services: By providing easier-to-access, more personalized services that can help individuals at risk better manage their own well-being, getting the right support when they need it. A specific use case is Aspiranet, which in 2020 served 22,000 youth and families across California, while using natural language inquiry of unstructured data to help youth transition from foster home care to living on their own; their CEO confirms that cognitive technology helps free up caseworkers time, enabling them to focus on what matters most, human connection. This is a significant impact given that in the United States, social worker turnover is as high as 90 percent per year, with heavy workloads, including paperwork, as a major contributing factor.

2) Providing environmental protections: IBM gives the example of their work with the Beijing Environmental Protection Bureau to reduce air pollution in China (which is the cause of more than one million premature deaths per year) via the 2014 Green Horizons initiative, which addresses these challenges by using advanced machine learning to identify smaller sections of the city that are at risk. Along with trade-off analyses, this enables more targeted mitigation actions, such as shutting down targeted industries, while minimizing socioeconomic disruptions.

3) Enhancing public safety: IBM gives the example of how they used Watson to anticipate and respond to the Oct 2015 Hurricane Patricia (one of the strongest storms ever recorded). In that case, weather prediction models based on artificial intelligence (AI) provided advanced warning to an IBM production center in Guadalajara. The system had analyzed huge volumes of disparate information, including weather data, social feeds and news reports to get a comprehensive view of the storm's trajectory. Early warning gave site officials the crucial time they needed to act and they opted to evacuate the site as a precautionary measure.

4) Personalized medicine and nutrition: Another example of social impacts that have been created through the adoption of MPJ's innovations is found in the work of ActiveEon. As mentioned earlier, ActiveEon has used the component-based MPJ development methodology in their ProActive Workflows and Scheduling software which incorporates dynamic and autonomous workload allocation to an elastic pool of resources using multi-language software components within the HPC cloud-based workflows. Among a range of case studies demonstrating usage of this software across sectors including mining, satellite technology, visa processing, and solvency solutions, is its use for microbiome analytics orchestration. The enabling of microbiome analytics orchestration at this scale is of significant benefit to the biomedical field as microbiome innovation encompasses pharmaceutical applications such as personalized medicine and nutrition, identification of the impact of drugs on the gut microbiome, and identification of bacterial biomarkers associated with dysbiosis (microbial imbalance).

V. Conclusions

In summary, MPJ's invention and results on optimizing the use of the Java programming language by introducing message passing in it has ushered in a new world of possibility and creativity for global industries and end users.

Acknowledgment

The pioneering research and development on MPJ have been one of the main areas of focus for the Distributed and Intelligent Systems Research Group (DIS RG) at the University of Westminster in London since the late 90s. Key members of this research team were: Alessandro Basso, Artie Basukoski, Alexander Bolotov, Jeeshan Gazi, Stavros Isaiadis, Sava Minchev, Jeyan Thiyagalingam, and Thomas Weigold. The early success of DIS RG attracted high international interest and the creation of the International MPJ Working Group with participation from the UK, USA, Europe, and Japan. Many colleagues from different countries around the world have provided significant contributions to the MPJ developments and success.

References

[1] M. Philippsen, R. Boisvert, V. Getov, R. Pozo, J. Moreira, D. Gannon, and G. Fox, "JavaGrande – high performance computing with Java," in Applied Parallel Computing, Lecture Notes in Computer Science, vol. 1947, pp. 20-36, 2001.

[2] B. Carpenter, V. Getov, G. Judd, A. Skjellum, and G. Fox, "MPJ: MPI-like message passing for Java," Concurrency: Practice and Experience, vol. 12 (11), pp. 1019-1038, Wiley, 2000.

[3] V. Getov, P. Gray, and V. Sunderam, "MPI and Java-MPI: Contrasts and comparisons of low-level communication performance," Proceedings of ACM/IEEE Supercomputing'99 Conference, pp. 1-21, IEEE, 1999.

[4] G. Judd, M. Clement, Q. Snell, and V. Getov, "Design issues for efficient implementation of MPI in Java," Proceedings of ACM 1999 Java Grande Conference, pp. 58-65, ACM Press, 1999.

[5] V. Getov, S. Flynn-Hummel, and S. Mintchev, "High-performance parallel programming in Java: Exploiting native libraries," Concurrency: Practice and Experience, vol. 10(11-13), pp. 863-872, Wiley, 1998.

[6] S. Mintchev and V. Getov, "Automatic binding of native scientific libraries to Java," Proceedings of ISCOPE'97, Lecture Notes in Computer Science, vol. 1343, pp. 129-136, Springer, 1997.

[7] B. Carpenter, V. Getov, G. Judd, T. Skjellum, and G. Fox, "MPI for Java: Position document and draft API specification," Technical Report JGF-TR-03, Java Grande Forum, 1998.

[8] O. Vega-Gisbert, J. E. Roman, and J. M. Squyres, "Design and implementation of Java bindings in Open MPI," Parallel Computing, vol. 59, pp. 1-20, 2016.

[9] K. Al-Attar; A. Shafi; H. Subramoni, and D. K. Panda, "Towards Java-based HPC using the MVAPICH2 library: Early experiences," 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 510-519, IEEE, 2022.

[10] F. Baude, D. Caromel, C. Dalmasso, M. Danelutto, V. Getov, L. Henrio, and C. Pérez, "GCM: A grid extension to Fractal for autonomous distributed components," Annals of Telecommunications, vol. 64(1-2), pp. 5-24, Springer, 2009.

[11] A. Basukoski, V. Getov, J. Thiyagalingam, and S. Isaiadis, "Component-based development environment for grid systems: Design and implementation," in Making Grids Work, pp. 119-128, Springer, 2008.

[12] T. Weigold, M. Aldinucci, M. Danelutto, and V. Getov, "Process-driven biometric identification by means of autonomic grid components," Int. Journal of Autonomous and Adaptive Communications Systems, vol. 5(3), pp. 274-291, Inderscience, 2012.

[13] V. Getov, "Component-oriented approaches for software development in the extreme-scale computing era," in High Performance Computing: From Grids and Clouds to Exascale, pp. 141-156, IOS Press, 2011.

[14] V. Getov, G. von Laszewski, M. Philippsen, and I. Foster, "Multi-paradigm communications in Java for grid computing," Communications of the ACM, vol. 44(10), pp. 118-125, ACM Press, 2001.

[15] R. Armstrong, G. Kumfert, L. McInnes, S. Parker, B. Allan, M. Sottile, T. Epperly, and T. Dahlgren, "The CCA component model for high-performance computing," Concurrency and Computation: Practice and Experience, vol. 18, pp. 215-229, Wiley, 2006.

[16] Open MPI: Open source high performance computing, accessed on June 7, 2023, https://www.open-mpi.org/.