published in Alexander Schill (Ed.): DCE - The OSF Distributed Computing Environment. International DCE Workshop. Proceedings, p. 39-46, Karlsruhe, Germany, Oct. 1993.

Comparison of DCE RPC, DFN-RPC, ONC and PVM

Rolf Rabenseifner[1] and Armin Schuch[2]

[1] Rechenzentrum der Universität Stuttgart, Allmandring 30, D-70550 Stuttgart, Germany, Tel. ++49 711 6855530, e-mail: rabenseifner@rus.uni-stuttgart.de
[2] Institut für Kernenergetik und Energiesysteme, Pfaffenwaldring 31, D-70550 Stuttgart, Germany, Tel.: ++49 711 6852122, e-mail: rpcas@ikeux1.energietechnik.uni-stuttgart.de

Abstract. Taking part in the Early Participation Program of OSF/DCE on IBM RS/6000 workstations, we have examined the RPC of DCE between workstation and compute server under aspects of performance, capability and functionality for scientific-technical applications programmed in Fortran, under user-account. A brief introduction shows the demands expected from a RPC tool taking a scientific-technical point of view.

1 Introduction

Taking a scientific-technical point of view, the demands, a RPC tool has to satisfy, are resulting from the profile of the user and from the application itself.

Users of scientific-technical applications are highly specialized, but they normally have only basic knowledge in computer science. Therefore, the user wants a short period of learning time to get his work done, this means the RPC tool must be easy to use. The lesser the difference between a distributed and a non-distributed application is, the better it is. The RPC tool should hide the distribution to the user as much as possible. The possibility of optimizations, which are only useful to users with good knowledge of the used tool, are not so important.

The features an application demands from a RPC tool can be shown looking at a typical, practical example. The program system ESTER has been developed for the shared-cost action for the reactor savety of the European Community [4, 12].

To harmonize the analyse of source terms beyond the countries of the EC, to strengthen the co-operation of single development groups and to develop a best-estimate code for source term analysis which can be used everywhere in Europe are goals of the ESTER development. This best-estimate code is based on modules which are developed at different research locations in countries which are members of the EC. The data exchange between these modules is made using a data base which is kept in the main memory because of efficiency reasons. These single modules are big program packages which were developed for specific computers. If you want to connect these modules, this can't be done anymore on a single computer. Therefore, communication between the modules and the data base with RPC tools is necessary.

The development of the modules has been done in projects over serveral years and the programming has been done exclusively in Fortran. At the moment, Fortran is a very important programming language for scientific-technical applications. So, a RPC tool must have a Fortran interface to satisfy these requirements. This interface must have the ability that remote procedures as Fortran subroutines and functions can be called from Fortran programs. Programming of C routines is undesirable.

The data exchange between the modules is made using a common data base which is kept in the main memory of any computer. When a calculation and afterwards a visualization is made where several modules take part, there are big data flows between the modules and the data base. Therefore, the ability of a RPC tool to use a high transmission speed is desirable.

In the summery, there are three demands a RPC tool has to satisfy: A RPC tool should be easy to use, should have a Fortran interface and should make it possible to use a high transmission speed.

2 Performance Comparison

To test the performance, we've compared the DCE RPC, IBM-Rel. 1.2.9310.0 (OSF-Release 1.0.1.) [8] with three other tools:

1. the DFN-RPC, Rel. 1.0, a RPC tool for the distribution of Fortran applications between workstation and compute server developed by us by order[3] of the DFN, the German Research Network Society [1, 9, 10, 11],
2. the PVM, Rel. 2.4.1 and 3.0, a message passing library to parallelize Fortran and C applications in a network of UNIX computers [2],
3. and the SUN RPC (ONC), a RPC tool for system programming [14].

For the first benchmark, shown in Table 1, two IBM RS/6000 workstations have been used, connected with Ethernet, FDDI, Cisco and NSC router. The RPC of DCE[4] and PVM are using UDP and an own protocol to do end-to-end flow control, while the DFN-RPC and the SUN RPC[5] are using TCP. The tests transmitted 1, 3 or 64 kbyte real numbers (each with 4 bytes length) as input or output arguments, the transmission speed was measured in kbyte/sec. The delay was measured by the round-trip-time of an empty call in ms/call.

The DFN-RPC was used with the default buffer size of 2400 bytes. The buffer size can be optimized between 600 bytes and 64 kbytes. Using PVM, the remote procedure call has been simulated by sending corresponding messages.

The results in per cent compare the speed of the DCE RPCs with the speed of DFN-RPCs, PVM and SUN RPCs. Looking at the empty calls, the per cent results are based on calls/ms. The more powerful CPU of the two CPUs taking

[3] registration number of the BMFT: TK 558 VA 005.3.
[4] In the next release, TCP will be available too.
[5] Also UDP is available, but only with may-be semantics, and therefore not evaluated in this test.

Table 1. Throughput and delay between two workstations

Calls with	Unit	DCE RPC		DFN-RPC		PVM 3.0		SUN RPC	
	abs.	abs.	%	abs.	%	abs.	%	abs.	%
64 kbyte In	kbyte/s	376	100	499	133	157	42	357	95
64 kbyte Out	kbyte/s	326	100	392	120	124	38	250	77
3 kbyte In	kbyte/s	141	100	196	139	82	58	134	95
3 kbyte Out	kbyte/s	140	100	124	86	78	56	117	84
1 kbyte In	kbyte/s	70	100	92	131	38	54	77	110
1 kbyte Out	kbyte/s	66	100	90	136	37	56	69	105
Average	kbyte/s	187	100	232	124	86	46	167	89
Empty Call	ms/Call	10.3	100	5.8	178	19.7	52	7.3	141

Table 2. Throughput and delay, if client and server process are on the same workstation

Calls with	Unit	DCE RPC		DFN-RPC		DFN 64k		PVM 3.0		SUN RPC	
	abs.	abs.	%	abs.	%	abs.	%	abs.	%	abs.	%
64 kbyte In	kbyte/s	159	100	1081	680	1336	840	286	180	331	208
64 kbyte Out	kbyte/s	203	100	1099	541	1481	730	286	141	356	175
3 kbyte In	kbyte/s	433	100	593	137	750	173	185	43	268	62
3 kbyte Out	kbyte/s	433	100	577	133	778	180	185	43	268	62
1 kbyte In	kbyte/s	156	100	366	235	376	241	88	56	145	93
1 kbyte Out	kbyte/s	163	100	355	218	385	236	84	52	165	101
Average	kbyte/s	258	100	679	263	851	330	186	72	256	99
Empty Call	ms/Call	6.0	100	2.4	250	2.3	261	9.0	67	3.8	158

part (the side of the compute server) was loaded 100% with a daemon. All measurements were repeated three times, and all three results have differed only a little bit. Therefore, the average results shown in the table can be used to compare the products.

DCE RPC and SUN RPC have nearly the same transmission speed. The DFN-RPC is faster by 25%, while PVM is slower by 50%. Empty calls are rather slow with DCE. DCE RPC needs 10.3 ms for a an empty call, this means 1.8 times longer than DFN-RPC, although it uses UDP. On the other hand, this can be the reason for the slowness, because the end-to-end flow control has to be made in the application process. Looking at PVM empty calls, you can see clearly that another two PVM deamons are involved.

The asymetry between In and Out measurements looking at the absolute numbers is probably caused by the different power of the computers. Another reason are the different routing paths through the network for both directions, while in both cases an Ethernet is the bottle-neck.

If client and server are on the same RS/6000 530, the differences are more visible, see Table 2. These measurements don't depend on a real network load and the computer has been used only for these measurements. Using DFN-RPC, two different buffer sizes have been used for the measurements, one with 2400 bytes (default) and one with 64 kbytes (64 kbytes is the recommendation for high speed networks >100 mbit/s). The buffer length can be changed by runtime, but can also be given in the interface definition file, used for the stub generation.

The DCE RPC shows a weakness if plenty of data has to be transferred. Other measurements on computers from Sun and Silicon Graphics [6] showed that SUN RPC has problems with a data quantity over 3.9 kbytes, which can cause in extreme cases a fixed round-trip-time of 200 ms and this means 19.5 kbytes/sec transmission speed for 3.9 kbytes of data, see also next section.

It's surprising that the performance of DCE, SUN RPC and PVM is so low in comparison with DFN-RPC, which uses internally only needed BSD socket system calls for a TCP connection, although the maximum network speed cannot be achieved with those sockets. To reach the maximum speed, more complex solutions are necessary, like the implementation in, e.g., the Peregrine High-performance RPC [3]. At least, the hope remains that a TCP based DCE RPC for applications under user account shows better performance and that, for kernel resident applications (like DFS), the UDP basis is not a disadvantage for the performance.

3 Possible Reasons for the Different Performance

The main topics of optimization of the DFN-RPC are:

- The DFN-RPC is making only an absolute minimal number of sytem calls. Only bsd socket `write` and `read` is done, using a TCP/IP connection, which is normally established the whole time during the execution of the program.
- The input and output arguments are copied only once within the user's space. They are copied between the argument list and an input or output buffer, which is used then for the `write` and `read` operations.
- The conversion – if necessary – is integrated into this copy and is done by loops with maximum length.
- These loops can be vectorized.
- The argument list on server side is allocated on a local stack and not by `malloc`.
- As transfer encoding of real and integer numbers one can choose between four formats: ieee big or little endian, cray or vax.

Because we have used the tools DCE RPC, SUN RPC and PVM only as a black box, it is difficult to say a lot about the reasons why these products don't reach the performance of the DFN-RPC. Nevertheless there are a few decisions in the design of those tools which must have a negative impact on the performance of these tools.

Table 3. In an empty Ethernet between two Silicon Graphics Indigo, IRIX 4.0.5F with E++, processor R 3000 and R 4000

Calls with	Unit	DFN-RPC		DFN 64k		PVM 3.0		SUN RPC	
	abs.	abs.	%	abs.	%	abs.	%	abs.	%
64 kbyte In	kbyte/s	800	100	663	83	286	36	487	61
64 kbyte Out	kbyte/s	699	100	753	108	312	45	499	71
3 kbyte In	kbyte/s	524	100	524	100	187	36	410	78
3 kbyte Out	kbyte/s	450	100	500	111	182	40	336	75
1 kbyte In	kbyte/s	294	100	300	102	92	31	238	81
1 kbyte Out	kbyte/s	294	100	307	104	87	30	195	66
Average	kbyte/s	510	100	508	100	191	37	361	71
Empty Call	ms/Call	2.1	100	1.9	111	8.6	24	2.3	91

Table 4. SUN RPC with 4 kbyte input or output data in an empty Ethernet between two Silicon Graphics Indigo, IRIX 4.0.5F with E++, processor R 3000 and R 4000

Calls with	Unit	DFN-RPC		DFN 64k		PVM 3.0		SUN RPC	
	abs.	abs.	%	abs.	%	abs.	%	abs.	%
4 kbyte In	kbyte/s	571	100	603	106	116	29	**20**	**3.5**
4 kbyte Out	kbyte/s	500	100	571	114	176	35	309	62

The DCE RPC in the tested release (the newest one that was delivered in the *Early Participation Program* of IBM in March 1993) is based only on UDP, which is also the default for system applications as the *distributed file system* (DFS). To obtain an at-most-once semantics and to guarantee the end-to-end-flow-contol, an additional protocol, executed by the RPC stubs, has to be implemented. This can lead to additional context switches between the application and the system kernel, which slows down the speed of data transfer and empty calls. I hope that this problem has no negative impact on RPC applications, like DFS, which are running in the kernel.

We have tested the SUN RPC also between two Silicon Graphics systems, see Tables 3–5. Looking at RPCs with 64 kbyte output data in Table 3 the DFN-RPC transfers 699 kbyte/s, i.e. it needs 5.6 μs for one real number. The SUN RPC transfers 499 kbyte/s, i.e. it needs 7.8 μs/number. Therefore, the loss of performance is 2.2 μs for each real number. The SUN RPC makes one conversion call for each real number (although this procedure makes only a copy). For the overhead of an additional procedure call on both systems we have measured 0.5 μs + 0.2 μs = 0.7 μs. This overhead is a significant part of the performance loss.

Table 4 shows a special problem of the SUN RPC. If one is using 4 kbyte of input arguments in a remote procedure call, then the call needs 200 ms with the consequence of a transfer rate of only 20 kbyte/s. This effect can be seen in a big

Table 5. Client and Server on the same Silicon Graphics with R 3000 processor

Calls with	Unit abs.	DFN-RPC abs.	%	DFN 64k abs.	%	PVM 2.4 abs.	%	PVM 3.0 abs.	%	SUN RPC abs.	%
64 kbyte In	kbyte/s	1012	100	1481	146	157	16	486	48	**320**	**32**
64 kbyte Out	kbyte/s	1040	100	1422	137	123	12	492	47	**320**	**31**
3 kbyte In	kbyte/s	705	100	868	123	105	15	331	47	450	64
3 kbyte Out	kbyte/s	727	100	938	129	91	13	326	45	450	62
1 kbyte In	kbyte/s	455	100	455	100	58	13	154	34	250	55
1 kbyte Out	kbyte/s	455	100	455	100	53	12	154	34	250	55
Average	kbyte/s	732	100	937	128	98	13	324	44	340	46
4 kbyte In	kbyte/s	800	100	1000	125	117	15	343	43	**20**	**2.5**
4 kbyte Out	kbyte/s	800	100	1000	125	100	13	333	42	**20**	**2.5**
Empty Call	ms/Call	1.4	100	1.4	100	11.4	10	4.9	29	2.1	67

Table 6. CPU time used in the experiment, shown in Table 2

Tool	process	CPU time	sum	comparison	*Average from Table 2*
DCE RPC	client	380 sec			
	server	391 sec	sum = 771 sec	= DCE-time / 1.00	*100 %*
DFN-RPC[6]	Server	168 sec			
	client	168 sec	sum = 336 sec	= DCE-time / 2.29	*263 %*
PVM 3.0	server	363 sec			
	client	362 sec			
	daemon	494 sec	sum = 1219 sec	= DCE-time / 0.63	*72 %*
SUN RPC	client	365 sec			
	server	391 sec	sum = 756 sec	= DCE-time / 1.02	*99 %*

range of input and output quantities larger than 3.88 kbyte, e.g. see values in Table 5 printed bold. We have tested it too on a SUN and have obtained similar results.

PVM is a message passing library. It was examined because it is also a tool to distribute scientific-technical applications. At the moment, PVM is outstanding because of its good functionality for parallel applications, but it is very slow, because it uses daemons for data transfer. Although there was an increase of performance of a factor of 3 between PVM 2.4.1 and PVM 3.0, as shown in Table 5, there must be done a lot to obtain full performance.

A further indicator is the consumption of cpu time, as shown in Table 6. Obviously, there is a correlation to the performance shown in Table 2.

[6] With a buffersize of 2400 bytes.

4 Functionality

Fortran application interfaces consisting of subroutines, entries, and functions of type INTEGER, REAL or DOUBLE with arguments of type INTEGER, REAL, DOUBLE, COMPLEX and arrays (with fixed lengths) of these types can be distributed without problems with DCE RPC. DCE RPC has the following problems with other language elements of Fortran [5]:

- The application interface (function names and argument list) defined in Fortran has to be redone in C, so that a corresponding Interface Definition File can be written as the input for the DCE RPC stub generator. The language transition from Fortran to C depends on the manufacturer, but it is documented in most cases, products as fidl [7] may also help here.
- If this manufacturer dependent language transition is different and therefore a different RPC protocol arises on client and server, the direct use of DCE RPC for Fortran applications isn't possible anymore. Examples for this are functions or arguments of type LOGICAL or CHARACTER. On computers with a different length of a *numerical unit* (e.g. workstation 4 byte, compute server 8 byte) the type REAL*8 (or equivalent Fortran 90 definitions) is changed to `double` in C on the workstation and to `float` in C on the compute server. In this case, DCE RPC doesn't find a proper network presentation between client and server, so this type can't be transmitted. This is also true for COMPLEX*16. On the contrary, different language transitions of the function names in Fortran to names in C aren't a problem, because only the position of the function in the IDL file is transmitted.
- For the Fortran language element *alternate return* no transition into C can be found and *function name arguments* are not supported by DCE RPC.

In these last two cases, the application programmer either has to change the Fortran interface or has to write additional stubs which put his interface into an interface that can be distributed with the DCE RPC.

Besides this, the possibilities with arrays that have variable dimensions and different or separated input and output areas are very limited. This can cause the situation that more elements than the application needs have to be transmitted.

Therefore, looking at functional aspects, in most cases, tools with a Fortran interface (e.g. DFN-RPC, PVM) are the better choice for distributing scientific-technical applications programmed in Fortran.

5 Conclusion

Looking at the three requirements, that such tools should be easy to use, should have a fortran interface and that they must be efficient, it is desirable that the DCE RPC gets a better performance without weakness in particular environments. It should get also a Fortran interface like the DFN-RPC and an additional Fortran message passing library like PVM.

References

1. Allrutz R., Rabenseifner R.: Der DFN-RPC, ein Remote Procedure Call Tool. Proceedings, 15. DECUS München Symposium (1992) 523–532
2. Geist, A. et al.: PVM 3.0 User's Guide and Reference Manual. ORNL/TM-12187, Oak Ridge National Laboratory, Tennessee (2/1993)
3. Johnson, D. B., W. Zwaenepoel: The Peregrine High-performance RPC System. Software-Practice and Experience **23(2)** (1993) 201–221
4. Jones, A. V., Shepard, I.: ESTER - a European Source Term Evaluation System. CEC Joint Research Centre, Safty Technology Institut (6/1991)
5. Kollak, W.: Distribute Fortran Applications using DCE RPC. White Paper. Rechenzentrum Universität Stuttgart, March 5, 1993.
6. Kollak, W., R. Rabenseifner, H.D. Reimann: RPC Tools im Benchmark-Vergleich: DFN-RPC, SUN-RPC, DCE RPC, sowie PVM. White Paper. Rechenzentrum Universität Stuttgart, 31.3.1993.
7. Laifer, R., A. Knocke: fidl, Ein Werkzeug zur einfachen Verteilung von Fortran-Anwendungen. Benutzeranleitung, Universität Karsruhe, Rechenzentrum, 31.3.1993.
8. Open Software Foundation (OSF): OSF DCE Version 1.0, DCE Application Development Guide, Part 3: DCE Remote Procedure Call (RPC). Revision 1.0, Dec. 31, 1991.
9. Rabenseifner, R., H.D. Reimann: Verteilte Anwendungen mit dem DFN-RPC. DFN Mitteilungen **31**, Berlin (3/1993)
10. Rabenseifner, R., H.D. Reimann: Verteilte Anwendungen über Hochgeschwindigkeitsdatenkommunikation, der DFN Remote Procedure Call. Benutzerhandbuch, Rel. 1.0, Rechenzentrum Universität Stuttgart, 23.11.1992.
11. Rabenseifner, R.: Distributed Applications between Workstation and Supercomputer using ISO/OSI Protocols. Proceedings, Twenty-Seventh Semi-Annual Cray User Group Meeting, London. April 22-26 (1991) 80–84
12. Schmidt, F., Schuch, A., Hinkelmann, M.: Der Europaeische Quellterm-Code ESTER – Grundideen und Werkzeuge zur Kopplung von ATHLET und ESTER. Abschlussbericht BMFT-Vorhaben 317-4015-1500945, Universität Stuttgart, **IKE 4-136** (4/1993)
13. Schuch, A.: Die Anwendung von Remote Procedure Calls für die Verteilung eines Anwendungsprogramms. Diplomarbeit. Institut für Kernenergetik und Energiesysteme, Universität Stuttgart (11/1991)
14. SUN microsystems: Network Programming Guide. Part Number 800-3850-00 (1990)