Joint WG5 J3 Meeting
February 2008

Dan Nagle
Chair J3

Introduction

To start, these are my personal views and not those of anyone else. I'm simply reporting how I saw things unfold. This is nothing more than my recollection. On the other hand, I'm writing this during the closing sessions of the February, 2008, meeting, so the events are fresh in my mind.

Background

The February, 2008, meeting was for WG5 to review J3's work of the several preceding years, work done at WG5's direction. J3 had been developing a new revision of Fortran, called Fortran 2008. The major feature of Fortran 2008 is parallel programming expressed by coarrays.

Coarrays

Coarrays are a parallel programming scheme where a program is imagined to exist as one or more instances of a serial program, each instance being called an image. Designated data items in one image may be referenced by another image. These designated items are called coarrays.

A coarray may be a scalar or an array, and may be of intrinsic type or derived type. The coarray attribute is indicated

in source code by the presence of square brackets following the variable name. Synchronization statements are provided so the programmer can synchronize the images.

Thus, coarrays are a concise, easy-to-understand, parallelization scheme. Communications between images occur only where square brackets are present, or where synchronization statements occur, and nowhere else. WG5 designated coarrays as a high-priority work item in 2005, at the joint meeting in Delft. This decision was confirmed at the February, 2006, meeting in Fairfax. J3 has spent the years since the Delft meeting writing coarrays into the draft of the Fortran 2008 standard. This was a significant effort, and consumed a great deal of J3's resources.

At the August, 2007, in London, WG5/J3 heard the first opposition to coarrays, in the form of abstentions to the resolutions of the meeting. This was J3's first indication that coarrays had weak support, or even opposition, within WG5. So the major issue to be decided at the February, 2008, joint meeting in Las Vegas was: Will coarrays be in the Fortran 2008 standard?

Two points of view emerged on the question. One view held that while parallel programming might well be important for the future, in the present, very few programmers actually use it. Further, while many parallel programming schemes have been tried over the years, most of them have failed for one reason or another. Thus, caution should be the guide. We should not commit coarrays to be in the standard. So, some non-committal means should be used to describe coarrays.

My views, which I believe are shared by many other members of the US delegation at the meeting, is that parallel programming is needed today; that the only commodity hardware available today and in the future has and will have multi-core, or greater, parallelism; and that portability of codes among various current and foreseeable hardware requires parallelism in the standard. Further, Fortran will have difficulty maintaining its market position if it cannot be used to effectively program commodity hardware by a highly portable means.

In many respects, this issue is a chicken-and-egg question. One side argued that if applications programmers want coarrays, they will demand them and vendors will provide them. Then, they can be included in the standard. The other side argued that without high confidence that coarrays will be supported by most compilers, such as can only come from coarrays being in the standard, few applications programmers will risk writing coarray codes. So failing to put coarrays in the standard will create a self-fulfilling prophesy that coarrays will not be used.

Background considerations should be mentioned. One reason so few parallel programs exist today is that parallel hardware in the past has been associated with more expensive, high performance hardware. But the present and future are clearly different. Parallel hardware is now and will continue to be ubiquitous. Check the advertisements for laptops, for instance.

A further consideration is the software. The only widespread parallel programming paradigm today is message passing. By widespread, I mean usable on most kinds of hardware, from SMP to DMP and the various combinations. Many times, message passing is implemented via MPI. MPI is terribly tedious and difficult to use, but message passing is the only way to write highly scalable programs today.

So many applications which could benefit from parallelization don't bother due to the degree of difficulty of using designing and writing message passing programs.

The other currently popular approach is OpenMP. The difficulty here is that it's an approach where compiler directives convert an otherwise correct serial program to a parallel program. This is an obsolete technology, first used (in my experience) in the 1980s. (The idea evolved from directives used for vector processing from the 1970s.) It is constrained in design by the need to keep the serial program semantics. It is constrained in practice to SMP hardware. It is more difficult to get right on ccNUMA hardware. However, it is much easier to use successfully than MPI. It just isn't highly scalable.

The story of HPF was cited by both sides of the debate. The anti-coarrys faction mentioned that HPF had been promoted as the parallelism of the future, yet it eventually failed. So we shouldn't make that mistake again. The pro-coarrays side mentioned that when HPF was proposed, hardware was in greater flux that at present, and that no vendor at the time had a full, highly optimizing HPF compiler. Today is different. Hardware is almost exclusively commodity hardware and one may expect it to remain so. Further, coarrays may be added to existing serial compilers while maintaining current optimization. HPF suffered, at least in part, because the subset actually supported by compilers varied from hardware to hardware (vendors supported only the subset that ran well on their hardware). Rightly or wrongly, HPF got the reputation of working well only on highly data-parallel problems. And the degree of difficulty of making good, complete HPF compilers greatly exceeded the original estimates. A fully standard and more general parallelism may be expected to fare better.

So why use coarrays? Coarrays have the advantage of running efficiently on hardware ranging from SMP to DMP and everything in between (for instance, DMP clusters of SMP nodes). Further, coarrays are a proven technology, having been used by Cray on several hardware platforms, for about a decade. Coarrays are understandable (compare with MPI) and yet run on highly parallel (100s to 1000s to 100000s of processors) (compare with OpenMP). And coarrays are simple, with serial images communicating at easily-seen points in the program.

The political positions appearing at the meeting divided according to whether coarrays should be a part of the main standard, or should be in a Technical Report, or should be a separate optional Part of the standard. I'll try to explain the standardise, but the underlying issue is how strongly to commit to coarrays in the standard.

The Fortran standard has used Technical Reports (TRS) previously. Both the Allocatable Components TR and the IEEE Arithmetic TR were issued as modifications to Fortran 95, and both were incorporated, with minimal modifications, into Fortran 2003. These earlier TRS were basically promises that the feature described would be incorporated as-is, with modifications only to correct errors found. There is currently an Enhanced Modules TR describing submodules to be included in Fortran 2008. J3 is currently working towards publishing a TR to describe how optional, allocatable, and assumed size arguments may be passed to programs written in C. This will become incorporated into a revision of the standard after Fortran 2008.

The Fortran standard has had separate Parts. These were the Varying Strings (Part 2) part, and the Conditional Compilation (Part 3) part. Both have been withdrawn. (The formal process will complete shortly but the Fortran committees' part is done or nearly so.)

The proposals to put coarrays in a TR were viewed as unacceptable to the US delegation. In major part, this is because the TR proposed had language that differed significantly from the other TRS. While the other TRS had been promises to include the features later, the language in the proposed coarrays TR would be only to review the coarrays TR later. Thus, there was much less confidence that coarrays would be included in the standard at all. I felt this would greatly inhibit applications programmers from wanting to use coarrays.

The proposal to put coarrays in a separate Part of the standard was likewise viewed as less than a promise to eventually put coarrays in the standard. The history of Parts of the Fortran standard does not inspire confidence, since all of them have been withdrawn. (In fairness, other languages maintain separate parts continually.)

During the course of the meeting, a number of compromises were proposed. One delegation proposed to allow the separate part of the standard containing coarrays to apply also to Fortran 2003 as well as Fortran 2008. This, it was argued, would speed acceptance of coarrays. The US delegation disagreed, because a vendor is always free to implement coarrays as soon as the definition is stabilized sufficiently to do so with confidence that the effort isn't wasted. (Any vendor must worry about the costs of their own efforts and those of their customers.) Thus, the compromise was more appearance than actual support for coarrays.

The US delegation proposed that the whole coarrays feature be separated into a kernel part to be kept in the standard, and a further features part that could be put in a TR to be included later. We called this compromise the "core and more" proposal. The "core" part is what is currently implemented by Cray and therefore represents a proven, workable subset of the original whole feature as defined by the draft. The "more" part were additions to the original coarrays added by J3 that experience with Cray's coarrays indicated would be useful to applications programmers. These features are unproven, and perhaps more work is needed. Thus, we felt that the concern of coarrays being unproven could be addressed by separating those as-yet unimplemented additions. These can get more work and further careful consideration.

WG5 operates by consensus, and after a few rounds of straw vote polling, the US proposal was found to be the proposal most likely to gain a consensus. Thus, it was adopted. The rest of the meeting was spent implementing that decision, and processing interpretations.

How could the process be improved? The most obvious answer is to have more participation in the standards process. Applications programmers and vendors both need to participate. Too many countries have only vendors or only applications programmers represented in the delegation. That's unbalanced. A per-country consensus is easier to achieve when broader views are present in each delegation.

Vendors will not support a new feature, unless they believe many applications programmers will want to use it. Not unnaturally, they want to make something that will make money. Applications programmers generally do not demand a new feature of a particular vendor unless they believe it will be generally available. Not unnaturally, they want to write portable code. So language evolution grinds to a halt as both parties wait for the other to move first.

Applications programmers, on the other hand, sometimes request features whose cost of implementation is high compared to the real value of the feature. That raises the cost of compilers without a commensurate benefit to applications programmers. That's part of what happened with HPF. So each party needs the other in order for the language to evolve in a long-term viable way.

In the end, I believe WG5 and J3 reached a good design for the future of Fortran, one which benefits and protects both applications programmers and vendors. So it turned out to everyone's benefit in the end.

Joint WG5 J3 Meeting February 2008

David Muxworthy
BSI Fortran Convener

Introduction

The principal business of the WG5 meeting in February was to review the content of the second Working Draft of the revised Fortran standard prior to its processing towards a CD. Many detailed clarifications and simplifications were discussed and approved, and a schedule for such processing was agreed. This should see the first public review take place during the summer with consideration of the public comments at the November 2008 meeting of WG5. The project is still on course for publication of the revised standard in June 2010.

At the previous meeting in August 2007 it had not been possible to achieve unanimity on the details of the coarray feature which introduces an SPMD (single program multiple data) model into Fortran to facilitate use of multi-processor systems. There are major differences of opinion on how Fortran should be adapted for multi-processor hardware. On the one hand there are those who think that Fortran should adopt the SPMD model whole-heartedly, who say that serial programming is dead and who think that Fortran could thereby regain some of the ground it has lost to other languages in the past ten to fifteen years. On the other hand there are those who think that some of the current parallel proposals have gone too far, that they are too complex as well as being experimental and untried, that they are expensive to implement and that they should not be imposed willy-nilly on vendors or users who might have no need of them. There is also a range of opinions between these two broad characterizations.

In an attempt to reach consensus, two discussion papers prior to the meeting had proposed (a) moving all coarray facilities to a Technical Report (TR), or (b) leaving a core facility in the base language and deleting the newer and so far unproven features, with their possible resurrection in a future revision. Debates on this subject occupied considerable time in the first three days of the meeting. An alternative proposal, and the only one acceptable to the Japanese member body, was that coarrays should be moved from the core language to a separate part of the standard, viz part 4. This would mean that the feature would be fully defined and standardized but that it would be optional for vendors to implement it. This was particularly relevant for Japanese vendors who are not convinced that coarrays are the correct model to adopt.

A series of straw votes showed that the meeting was in favour of certain reductions in coarray facilities, was evenly split on moving coarrays to a TR but was less keen on a move to a part 4. This caused the two larger delegations to go away and each come up with a new compromise plan. That for the US involved reducing the facilities in the base language and moving other features to a TR. The UK proposed that coarrays be moved en bloc to a part 4 of the standard and that document be written so that it could be treated as an addendum to both Fortran 2003 and Fortran 2008. Both plans allowed some decoupling of the development timetable of coarrays from that of the base language, giving more time to consider the as yet unproven coarray features.

A further set of votes, counted both by individual and by country, decided that the US plan was preferable. Hence this policy was followed for the remainder of the meeting and was endorsed in the resolutions. The detailed voting figures are shown in the minutes at ftp://ftp.nag.co.uk/sc22wg5/N1701-N1750/N1715.txt and the resolutions are in N1714; the schedule for the public consultation is in N1719. A new work item proposal for the TR will be forwarded to SC22 in due course.

Commentary

Opposition to coarrays has sometimes been characterized as the UK position but this is over-simplification. Some members of the UK Fortran group are happy to have some or all of the coarray features in the base language while some want them to be an optional feature; no-one in the UK group is saying that coarrays should be abandoned altogether. This reflects similar differences amongst UK Fortran users, some of whom are keen to access coarrays as soon as possible whilst others are largely indifferent and a few are actively opposed to their being in the core language on the grounds that the language could then be left with expensive redundant features if or when an alternative parallel model is implemented.

It was said to be ironic that it was the UK that proposed coarrays in the first place. This reminded me of the notices above the waste bins in RAF cookhouses fifty years ago, "Why did you take it if you didn't want it?" to which the obvious answer was, "We didn't know what it was going to be like".

It seems to me that WG5 has chosen the wrong option. The UK compromise would have given everyone what they wanted: a fully standardized coarray facility, no uncertainty about the status of the material in a TR and the ability of vendors to opt out of coarrays if they wished. It would have left the core language free of the coarray syntax which some fear may have limited useful shelf-life. It would have allowed users in institutions with standards-only policies access to all the facilities of coarrays instead of cutting out those which were only in a TR. It could even have seen coarrays implemented more quickly, as an addition to Fortran 2003 processors, since vendors with interested customers could then have developed that facility before starting on F08. Politically, it would have changed the Japanese vote from 'no' to 'yes' and the UK vote from 'abstain' to 'yes' at this meeting and possibly at the more formal SC22 votes to come.

The plan apparently failed largely because of a widespread misunderstanding of the status of a part 4. This was no doubt due to people thinking in terms of the unloved parts 2 and 3 of Fortran and not being aware of other ISO standards. Part 4 was described as a "lesser form of standardization"; this is not true - it would have had the same status as part 1. It was described as showing a "lack of commitment to parallelism", but it should not be the role of WG5 to try to lead the world's Fortran users into using parallel techniques, whether or not it is relevant their circumstances. Further, it was disappointing to see the serious objections of the Japanese vendors so lightly dismissed.

It is the role of WG5 to facilitate the provision of tools with which Fortran users can work on whatever hardware is available, and if users of other languages are thereby attracted to using Fortran, so much the better. Unfortunately WG5 has now chosen to follow a sub-optimal path.

C'est la vie.

Fortran Standard Activities

Organization of Standards Committees

J3 is the technical committee for Fortran standards development accredited by NCITS (National Council for Information Technology Standards). Information concerning membership in J3 can be obtained from the J3 Chairman (see listing below). Working Group 5 (officially ISO/IEC JTC1/SC22/WG5) has an analogous role in the International Standards Organization. J3 members and observers pay an annual participation fee; request further information from the NCITS Secretariat.

J3 Officers

Chairman: Dan Nagle (*dnagle@erols.com*)

Head of the Interpretations subgroup: Stan Whitlock *(Stan.Whitlock@intel.com)*

Host: Van Snyder (*van.snyder@jpl.nasa.gov*)

International Representative: Van Snyder

Secretary: Stan Whitlock

Treasurer: Van Snyder

J3 World Wide Web site

http://www.j3-fortran.org

NCITS Secretariat

(Information Technology Industry Council)

Secretariat,

ITIC

Suite 200

1250 Eye Street NW

Washington DC 20005

Tel: (202) 737-8888

Working Group 5 Convener

John Reid (j.reid@rl.ac.uk*)*.

J3 Meetings

J3 normally plans four meetings per year, scheduled for the second week of February, May, August, and November. See also International Meetings, below. Meetings of J3 are open to the public, but facilities are limited. If you would like to attend a meeting, request further information from Van Snyder at the address given above. The following was the last meeting scheduled:

183 10-15 February , 2008, Las Vegas, Nevada.

184 12-16 May, 2008 Las Vegas, Nevada.

185 16-21 November, 2008, Tokyo, Japan.

International Meetings

The international technical group for Fortran standardization is known as ISO/IEC JTC1/SC22/WG5; i.e., Working Group 5 (WG5) of Subcommittee 22 (S 22) of Joint Technical Committee 1 (JTC1) of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). This group usually meets each summer, and will next meet in August 2007 in the UK. The WG5 Web site is

http://www.nag.co.uk/sc22wg5/

A Reminder of the Schedule

Joint WG5 and J3 Meeting,10-15 February 2008, Las Vegas, Nevada.

Joint WG5 and J3 Meeting, 16-21 November, 2008, Tokyo, Japan.

join today!

SIGPLAN & ACM

www.acm.org/sigplan www.acm.org

The **ACM Special Interest Group on Programming Languages** (SIGPLAN) explores programming language concepts and tools, focusing on design, implementation, and efficient use. Its members are programming language users, developers, theoreticians, researchers, and educators. The monthly newsletter, ACM SIGPLAN Notices, publishes several conference proceedings issues, regular columns and technical correspondence (available in electronic or hardcopy versions). Members also receive a CD containing the prior year conference proceedings and newsletter issues. SIGPLAN sponsors several annual conferences including OOPSLA, PLDI, POPL and ICFP, plus a number of other conferences and workshops.

The **Association for Computing Machinery** (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to *Communications of the ACM*, *MemberNet*, *TechNews* and *CareerNews*, plus full access to the *Guide to Computing Literature*, thousands of free online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

❏ SIGPLAN Print (ACM Member or Non-ACM Member)	$ 65
❏ SIGPLAN Print (ACM Student Member)	$ 40
❏ SIGPLAN Online (ACM Member or Non-ACM Member)	$ 25
❏ SIGPLAN Online (ACM Student Member)	$ 15
❏ ACM Professional Membership ($99) & SIGPLAN Print ($65)	$164
❏ ACM Professional Membership ($99) & SIGPLAN Print ($65) & ACM Digital Library ($99)	$263
❏ ACM Professional Membership ($99) & SIGPLAN Online ($25)	$124
❏ ACM Professional Membership ($99) & SIGPLAN Online ($25) & ACM Digital Library ($99)	$223
❏ ACM Student Membership ($19) & SIGPLAN Print ($40)	$ 59
❏ ACM Student Membership ($19) & SIGPLAN Online ($15)	$ 34
❏ *SIGPLAN Notices* only (available in electronic or hardcopy versions)	$ 75
❏ *SIGPLAN Notices* Expedited Air (outside N. America)	$ 55
❏ *FORTRAN Forum* (ACM or SIGPLAN Member)	$ 10
❏ *FORTRAN Forum* (ACM Student Member)	$ 6
❏ *FORTRAN Forum* only	$ 20
❏ Expedited Air for *FORTRAN Forum* (outside N. America)	$ 10
❏ Expedited Air for *Communications of the ACM* (outside N. America)	$ 39

payment information

Name ____________________

ACM Member # ____________________

Mailing Address ____________________

____

City/State/Province ____________________

ZIP/Postal Code/Country ____________________

Email ____________________

Fax ____________________

Credit Card Type: ❏ AMEX ❏ VISA ❏ MC

Credit Card # ____________________

Exp. Date ____________________

Signature ____________________

Make check or money order payable to ACM, Inc

ACM accepts U.S. dollars or equivalent in foreign currency. Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Mailing List Restriction
ACM occasionally makes its mailing list available to computer-related organizations, educational institutions and sister societies. All email addresses remain strictly confidential. Check one of the following if you wish to restrict the use of your name:

❏ ACM announcements only
❏ ACM and other sister society announcements
❏ ACM subscription and renewal notices only

Questions? Contact:
ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
email: acmhelp@acm.org

Remit to:
ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

SIGAPP28

www.acm.org/joinsigs

Association for Computing Machinery

Advancing Computing as a Science & Profession