Литература

1. *Легалов А.И., Кузьмин Д.А., Казаков Ф.А., Привалихин Д.В.* На пути к переносимым параллельным программам // Открытые системы, 2003. № 5 (май). С. 36-42.

2. *Легалов А.И.* Инструментальная поддержка процесса разработки эволюционно расширяемых параллельных программ // Проблемы информатизации региона. ПИР-2003/ Материалы 8-й Всероссийской научно-практической конференции. Красноярск, 2003. С. 132-136.

3. *Маурер У.* Введение в программирование на языке ЛИСП // М.: Мир, 1976. - 104 с.

РЕАЛИЗАЦИЯ ФУНКЦИЙ СТАНДАРТА MPI ДЛЯ ЭМУЛЯЦИИ ОБМЕНОВ СООБЩЕНИЯМИ МЕЖДУ УЗЛАМИ МНОГОПРОЦЕССОРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

А.В. Лепихов

Челябинский государственный университет, г. Челябинск

В настоящее время вычислительные комплексы с массовым параллелизмом широко применяются для решения большого класса задач. Однако при использовании данных систем возникает сложная проблема отладки параллельных программ, не решенная в полной мере до настоящего момента.

Как известно отладка программы отнимает порядка 50% времени создания программы, а зачастую оказывается очень продолжительной. Сложность параллельных программ как таковых и их недетерминированное поведение превращают отладку параллельных программ в очень сложный для разработчика процесс.

Помимо приобретения коммерческого отладчика (например, TotalView [http://www.etnus.com] и PGDBG [http://www.pgroup.com]) существуют другие способы отслеживания событий программы при помощи вывода сообщений трассировки, распечатка значений переменных в заданном процессе и т. д. – весьма трудоемкий процесс. Предложенное решение проблемы выбора средств отладки основано на *эмуляции* параллельных процессов и обменов данными между ними с помощью стандартных средств ОС Windows.

Разработанный *эмулятор обменов сообщениями* представляет собой динамически компонуемую библиотеку, интерфейс которой есть подмножество функций стандарта MPI, реализующих асинхронный обмен сообщениями между процессами. Таким образом, эмулятор позволяет запускать параллельные программы непосредственно из среды

MS Visual C++ (без загрузчика) и использовать весь спектр средств встроенного отладчика.

Эмулятор обеспечивает представление процессов, запускаемых на процессорных узлах, в виде процессов ОС Windows, каждый из которых представляет собой совокупность следующих *потоков* (нитей): мастер, отправитель, получатель и терминатор. *Поток-мастер* выполняет собственно код процесса. *Поток-отправитель* обрабатывает массив, элементами которого являются очереди сообщений, передаваемых другим процессам программы. *Поток-получатель* обрабатывает очередь сообщений, поступающих от потоков-отправителей других процессов программы. *Поток-терминатор* выполняет аварийное завершение всех процессов программы, если поток-мастер одного из процессов выполнил функцию MPI_Abort.

Прием-передача сообщения от процесса *S* процессу *R* выглядит следующим образом (далее мы будем именовать потоки как *Master*, *Sender* и *Receiver*, а индексы имен будут указывать на принадлежность к процессу).

При выполнении функции посылки сообщения поток $Master_S$ добавляет в соответствующую очередь потока $Sender_S$ запись о данном сообщении и открывает ему семафор для начала передачи сообщения.

Поток $Sender_S$ создает в оперативной памяти процесса *S* область, доступную для потока $Receiver_R$, записывает в нее передаваемое сообщение и генерирует для $Receiver_R$ событие о необходимости начать прием. После этого поток $Sender_S$ переходит к ожиданию подтверждения от потока $Receiver_R$ о завершении приема. При получении подтверждения $Sender_S$ уничтожает ранее созданную область и памяти и закрывает семафор передачи сообщения.

Поток $Receiver_R$ выполняет перманентное ожидание события о необходимости начать прием. При наступлении такого события он обращается к области памяти, которую создал $Sender_S$, и считывает переданное им сообщение. После этого $Receiver_R$ высылает потоку $Sender_S$ подтверждение о завершении приема сообщения.

Разработанный эмулятор может быть использован для отладки параллельных программ, выполняющих обмен сообщениями на основе стандарта MPI. Переход от отладки к тестированию и обратно требует внесения минимальных изменений в исходные тексты, не влияющие на её работу с использованием стандартных реализаций MPI и изменения параметров их компоновки.