**Технология PARIX**

**Описание:**

PARIX (Parallel extensions to UNIX) является надстройкой над ОС AIX. При обращении к ОС PARIX, последняя захватывает ресурсы всех процессоров в указанном разделе и запускает на них вашу прикладную задачу.

PARIX - это однозадачная система: если один пользователь запустил PARIX-приложение на разделе, то никакой другой пользователь уже не сможет запустить своего PARIX-приложения на данном разделе, покуда первое приложение не закончит свою работу и не освободит раздел.

С одной стороны, такой подход несомненно ограничивает пользователей, заставляя их строго координировать свои расчетные задачи. С другой же стороны, такая схема обеспечивает наиболее благоприятный режим для запущенного приложения.

В PARIX принята следующая модель представления многопроцессорной системы: все процессоры “заполняют” ячейки в некоем трехмерном “ящике”. Каждому процессору приписаны уникальные координаты (тройка целых положительных чисел). Так что с точки зрения пользователя процессор кроме своего порядкового номера имеет “координаты”.

В трехмерном ящике с узлами не может быть дыр: если существует процессор с координатами (0,0,0) и существует (0,0,2), значит обязан существовать и (0,0,1). Кстати, узел (0,0,0) существует всегда (если в системе ненулевое количество процессоров).

Конечная трехмерная решетка, включающая в себя все процессоры данной машины, называется основным разделом. Обычно основной раздел делится на несколько более мелких (и, возможно, перекрывающихся) разделов. Самым мелким, понятно, является раздел, состоящий из одного узла. Важнейшей характеристикой раздела является его размер, т.е. размер трехмерного ящика с узлами, входящими в данный раздел.

**Топологии процессоров:**

В PARIX имеются синхронные и асинхронные операции обмена сообщениями, работающие с понятием виртуальной топологии - топологии сети, образуемой виртуальными каналами. PARIX предлагает богатый набор типов топологий - конвейер (Pipe), кольцо (Ring), двух/трех (2D/3D) мерные решетки (Grid) и торы (Torus), гиперкуб (HCube), клика (Clique), дерево (Tree и RTree). Более того, пользователь может создавать свои собственные топологии сетей, образованных виртуальными каналами.

Для каждого стандартного типа XXX топологии имеются функции, обеспечивающие создание, удаление коммуникационной сети с соответствующей топологией XXX.

int MakeXXX (int RequestId,

dims . . .,

int xmin, int xmax,

int ymin, int ymax,

int zmin, int zmax),

int FreeXXX (int topId),

XXXData\_t \* GetXXX\_Data(int topId)

MakeXXX создает коммуникационную сеть с соответствующей топологией XXX размерностью dim, которая отображается на подмножество процессоров, описываемое параметрами xmin, xmax, ymin, ymax, zmin. Только процессоры, имеющие в разделе координаты (x, y, z) такие, что xmin <= x <= xmax, ymin <= y <= ymax и zmin <= z <= zmax будут использоваться в этой сети. Сети с топологией XXX (а именно ее каналам) необходимо сопоставить уникальный идентификатор канала, передаваемый через параметр RequestId и используемый ОС для того, чтобы различать сообщения, передаваемые по каналам разных сетей. Функция возвращает идентификатор сети (топологии). Каждый узел сети имеет каналы, связывающие его с соседними узлами. Количество каналов зависит от вида топологии сети. Для указания каналов используются целые значения от 0 до <число соседних узлов>-1, называемые логическими каналами. Например, все узлы сети с топологией кольцо имеют два канала для двух направлений - вперед, назад, которым сопоставлены логические каналы 1 и 0 соответственно. В сети с топологией клика каждый узел имеет каналы, связывающие его со всеми остальными узлами сети, - логический канал i представляет виртуальный канал, ведущий из узла с номером n к узлу с номером, получающимся из x инвертированием i-го бита (x ^ (1<<i)).

Функция FreeXXX удаляет сеть, имеющую указанный идентификатор. Функция GetXXX\_Data возвращает указатель на дескриптор сети с указанным идентификатором. Дескриптор имеет следующий вид:

struct XXXData\_t {

char type; /\* содержит имя топологии вида: XXX\_TYPE (например. RING\_TYPE) \*/

int status; /\* указывает статус процессора, характеризующий его отношение к сети, его положение в сети. Например, для конвейера это значения PIPE\_HEAD, PIPE\_IN и PIPE\_TAIL. Если процессор не входит в соответствующую сеть, то он имеет статус XXX\_NONE (например. PIPE\_NONE). \*/

xxx\_t id; /\* идентифицирует положение процессора в топологии. Например, для конвейера это значение в диапазоне 0 .. <длина\_конвейера>-1, причем процессор со статусом PIPE\_HEAD имеет id =0 , а процессор со статусом PIPE\_TAIL - id = <длина\_конвейера>-1 \*/

xxx\_t dims; /\* хранит размерности топологии. Например, для 2D решетки это ее ширина и высота \*/

};

Функции

int Send (int TopId, int LogLinkId, void \*Data, int Size)

int Recv (int TopId, int LogLinkId, void \*Data, int Size)

реализуют синхронный обмен сообщениями в сети с топологией, имеющей идентификатор TopId. Параметр LogLinkId указывает, какой логический канал топологии следует использовать для приема/передачи сообщения.

Используя функции

int NewTop (int nLinks),

int AddTop (int TopId, LinkCB\_t \*Link),

пользователь может создавать свои собственные топологии сетей виртуальных каналов. Функция NewTop, с вызова которой начинается процедура формирования топологии, создает в памяти вызвавшего ее процессора некоторую внутреннюю таблицу под предполагаемое количество (nLinks) каналов, и возвращает идентификатор топологии. Идентификатор с этого момента представляет вновь образованную топологию. Внутренняя таблица будет хранить указатели на дескрипторы виртуальных каналов топологии.

Функция AddTop добавляет новый виртуальный канал Link к топологии TopId и возвращает идентификатор представляющего его логического канала (LogLinkId). С этого момента логический канал LinkId топологии TopId можно использовать для обмена сообщениями. Функция

LinkCB\_t \*GetLinkCB (int TopId, int LogLinkId, int \*Error)

позволяет получить указатель на виртуальный канал логического канала LinkId топологии TopId. Удалить топологию, е§ каналы можно с помощью функции

int FreeTop (int TopId),

которая удаляет внутренние структуры топологии, неявно освобождает каждый канал сети и внутренние данные, используемые при асинхронных коммуникациях.

Каждая топология имеет свои ей присущие свойства такие, как размерности dim\_x, dim\_y, dim\_z трехмерных решётки и тора, глубина дерева и число потомком у его вершин. Функция

int AddTop\_Data (int TopId, void \*Data)

позволяет связать с топологией её атрибуты, хранящиеся в дескрипторе Data. Затем их можно будет извлечь с помощью функции

void \*GetTopData (int TopId, int \*Error)

Следующий набор функций, реализует асинхронные обмены сообщениями по каналам виртуальных топологий.

int AInit (int TopId, int Threads, int Size)

int ASend (int TopId, int LogLinkId, byte \*Data, int Size, int \*Result)

int ARecv (int TopId, int LogLinkId, byte \*Data, int Size, int \*Result)

int ASync (int TopId, int LogLinkId)

int AExit (int TopId)

**Каналы связей:**

В PARIX введено понятие виртуального канала. Процессор может иметь необходимое число виртуальных каналов, которые могут связывать его с произвольными процессорами, расположенными в произвольном месте сети. PARIX сам осуществляет рассылку между транспьютерами сообщений, передаваемых по виртуальным каналам. Можно создать виртуальные каналы между транспьютером и внешним компьютером.

Чтобы создать виртуальный канал, два процесса, которые хотят установить связь с помощью этого канала, должны вызвать функцию:

LinkCB\_t \*ConnectLink(int Processor, int RequestId, int \*Error).

Каждый процесс, используя параметр Processor, должен указать номер процессора, с процессом которого он хочет установить связь. Каналу необходимо сопоставить уникальный идентификатор канала, передаваемый через параметр RequestId. Это значение используется ОС для того, чтобы различать сообщения, передаваемые по разным каналам. Функция возвращает указатель на дескриптор созданного канала.

Функция

int BreakLink (LinkCB\_t \*Link)

удаляет виртуальный канал. Каждый из процессов партнеров должен удалить свою часть канала.

Пересылка сообщений по раннее созданному каналу осуществляется с помощью синхронных операций:

int SendLink (LinkCB\_t \*Link, void \*Buf, int Size),

int RecvLink (LinkCB\_t \*Link, void \*Buf, int Size)

**Обмен данными:**

Осуществляется с помощью каналов, виртуальных топологий или используя синхронные операции обмена:

int SendNode (int Processor, int RequestId, byte \*Buffer, int Size) и

int RecvNode (int Processor, int RequestId, byte \*Buffer, int Size),

позволяют обмениваться сообщениями между процессами без создания виртуальных каналов. Для каждой операции каждый процесс-партнер, используя параметр Processor, должен указать номер процессора, процессу которого он хочет передать сообщение или принять сообщение от него. Операции обмена необходимо сопоставить уникальный идентификатор запроса, передаваемый через параметр RequestId. Значение -1, используемое в качестве Processor или RequestId, позволяет RecvNode получать сообщения от произвольного процессора или с произвольным идентификатором запроса.