Ефремов Н.Ф., к.т.н., доцент,
Ефремов Ю.Н., к.т.н., доцент,
Житомирский государственный технологический университет

НОВОЕ В ПРОГРАММИРОВАНИИ

В связи с переводом процессоров на многоядерную технологию, потребовало отхода от традиционного программирования, требующего по иншему мыслить, более обширно. Учитывать фактор многопоточности при работе с распределенными и параллельными системами с использованием многоядерных процессоров. При программировании возникает много трудностей из-за параллельности, это требует перестройки нашего сознания, привычек. В докладе выступления этим поднимаемым вопросам будет уделено больше внимания и дана оценка.

Многоядерные процессоры в одном корпусе выполняют вычисления, при многопоточности предоставляя доступ к общей памяти и к общим внешним шинам и сигналам. Это разширяет возможности компактности и приближения к аналоговым машинам.

Целью доклада переход к многопотоковому программированию, на базе многопроцессорных и многоядерных систем.

Переход к новой архитектуре многопроцессорных и многоядерных систем связан с тем, что физический предел частоты работы процессоров достигнут, а требования к быстродействию беспрерывно возростает. Все это потребовало поиска новых путей увеличения быстродействия.

В свете сказанного, необходимо отметить, что в настоящее время много сделано в направлении многопоточности, но нет еще достаточности понимания. Большие сложноси возникают с распараллеливанием алгоритмов, отсутствуем методик их создания. Использование графов для распараллеливания многопроцессорных и многоядерных систем, на нашу мысль, пока сложно и малоэффективно. Требует высокую квалификацию от разработчика.

В настоящее время перед разработчиками программного обеспечения, математиками поставлена задача поиск путей по созданию методик для распараллеливания многопроцессорных и многоядерных систем. В этом направлении трудятся ученые Украины.

Современная разработка архитектуры с распределенными и параллельными системами с использованием многоядерных процессоров, требует обучения студентов не на старших курсах, а начиная с младщих. Наше мнение такое, что нужно отказаться от обучения студентов традиционному программированию и переходить к распределенным системам включая параллельные системами на базе многоядерных процессоров, как можно раньше, это позволит, более плавно освоить новое направление в программировании.

Уделим внимание альтернативным вариантам повышения вычислительных мощностей компьютеров. Как мы знаем, что распределенные системы, состоящие из компьютеров, кластеры параллельных вычислений, в которых совместно работают несколько компьютеров, и многопроцессорные системы, где на одной материнской плате расположено несколько процессоров.

Оказавается, что чем больше процессоров, тем значительнее вычислительные мощности. Вычислительнаяй система увеличиваеть вычислительную мощность с ростом числа процессоров называется *масштабируемостью* системы. Построение масштабируемых систем является сложной и не всегда с получаемым решением.

Вычислительные примеры, требующие большой процессорной мощности, можно разделить на два направления: это примеры (задачи), решаемые с помощью не связанных друг с другом вычислений и это примеры решаемые с помощью связанных друг с другом вычислений, следует учитывать, что возникают ситуации, когда один или несколько вычислительных потоков вынуждены ждать другой поток до тех пор, пока он не поставит им необходимые данные. При обсуждении параллельных вычислений, имеются в виду параллельные алгоритмы обработки данных. Это совсем не обязательно численные расчеты.

Итак, многоядерная архитектура процессоров — единственно возможное решение, позволяющее повысить быстродействие современных процессоров для всех категорий примеров (задач) данным.

Таким образом, многоядерная архитектура в ближайшие годы позволит заметно увеличить производительность компьютеров, но только в том случае, если программное обеспечение даст возможность эффективно распараллелить вычисления.

Выводы. Программирование в многопоточной среде – это сложное и ответственное занятие, требующее очень высокой квалификации. Многие алгоритмы, кажущиеся простыми, естественными и надежными, в многопоточной среде оказываются неработоспособными, из-за чего способы решения даже самых простых примеров становятся необычными и запутанными, не говоря уже о проблемах, возникающих при решении сложных задач. В связи с этим вначале не следует увлекаться многопоточностью, нужно начинать с простого примера.