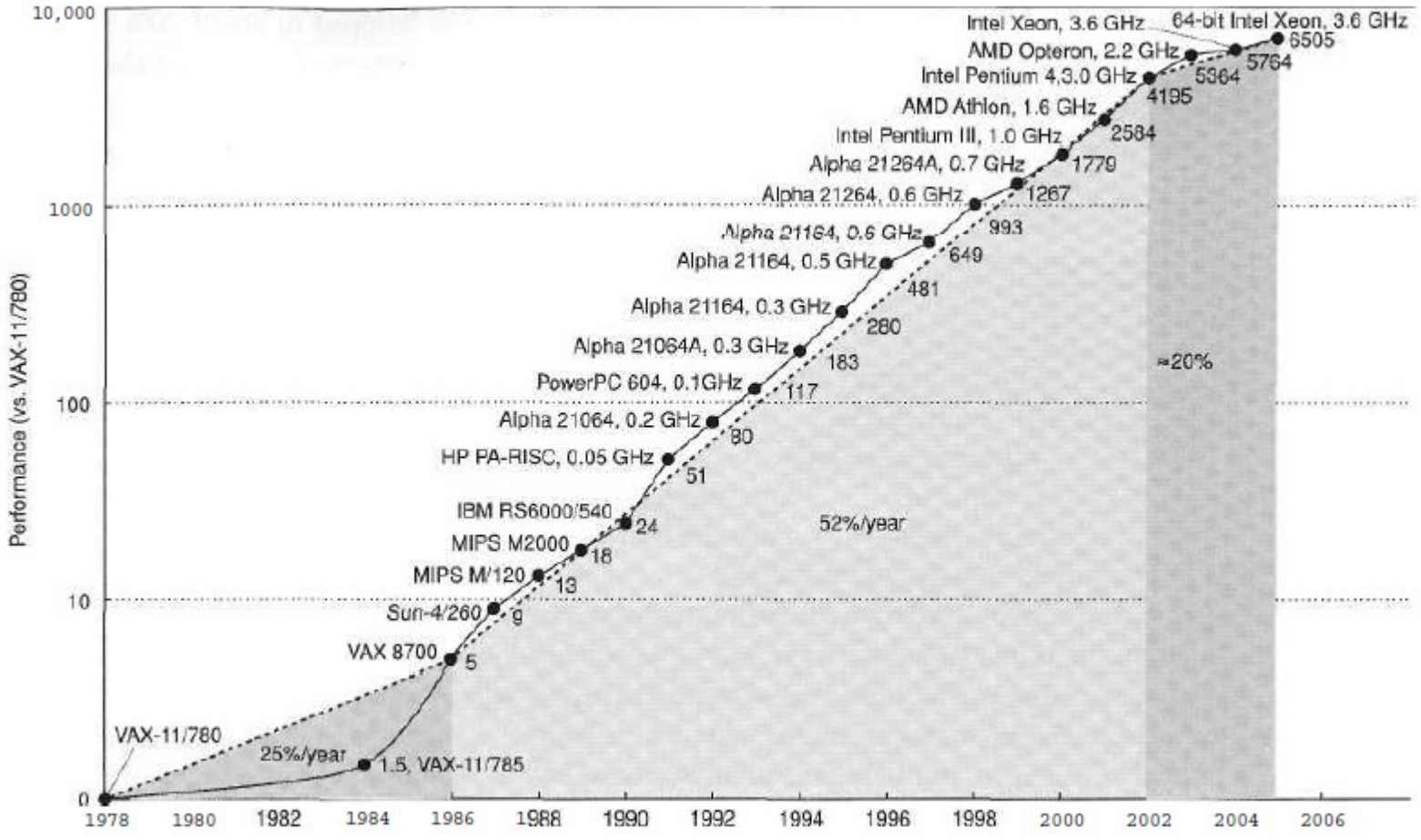
Intel Cluster OpenMP: параллельные программы для общей памяти могут работать на кластере

Александр Коновалов

Рост производительности процессоров, измеренный тестом SPECint



Из 4-го издания книги Hennessy, Patterson “Computer Architecture: A Quantitative Approach”

omp paralel for: простой и с мыслями

```
void for_2 (float a[], float b[], float c[], float d[], int n, int m)
{
      int i, j;
#pragma omp parallel for shared(a,b,c,d,n,m) private(i,j)
      for(i = 1; i < n; i++)
      {
         for(j = 0; j <= i; j++)
            b[j + n*i] = ( a[j + n*i] + a[j + n*(i-1)] )/2.0;
      }

#pragma omp parallel for shared(a,b,c,d,n,m) private(i,j)
      for(i = 1; i < m; i++)
      {
         for(j = 0; j <= i; j++)
            d[j + m*i] = ( c[j + m*i] + c[j + m*(i-1)] )/2.0;
      }
}
```

```
void for_2 (float a[], float b[], float c[], float d[], int n, int
      m)
{
      int i, j;
#pragma omp parallel shared(a,b,c,d,n,m) private(i,j)
      {
 #pragma omp for schedule(guided,1) nowait
        for(i = 1; i < n; i++)
        {
          for(j = 0; j <= i; j++)
             b[j + n*i] = (a[j + n*i] + a[j + n*(i-1)] )/2.0;
         }

 #pragma omp for schedule(guided,1) nowait
             for(i = 1; i < m; i++)
             {
                for(j = 0; j <= i; j++)
                 d[j + m*i] = ( c[j + m*i] + c[j + m*(i-1)] )/2.0;
             }
      }
}
```

Независимость итераций – это хорошо, но мало…

```
for (i=0; i<size; i++)
        if (global > arr[i])
            global = arr[i];
```

```
#pragma omp parallel
    {
     int my = omp_get_thread_num();
     int curr = INT_MAX;

     for (i=my*(size/all); i<(my+1)*(size/all); i++)
         if (curr > arr[i])
             curr = arr[i];
     if (curr < global) {
#pragma omp critical
             if (curr < global)
                     global = curr;
             }
    }
```

Состояние и перспективы OpenMP

- С появлением в GCC и Микрософтовских компиляторах проблема «это не библиотека, а язык» стала менее значимой.
- В настоящее время делается попытка уйти от слишком сильной связи с распараллеливанием циклов по массивам. Модельная задача – parallel_while:

```
while ( curr ) {
    if (foo(curr)) break;
    curr = curr->next;
}
```

- А появится ли в результате новое качество? Часто ли интересно ускорение в 4 раза?

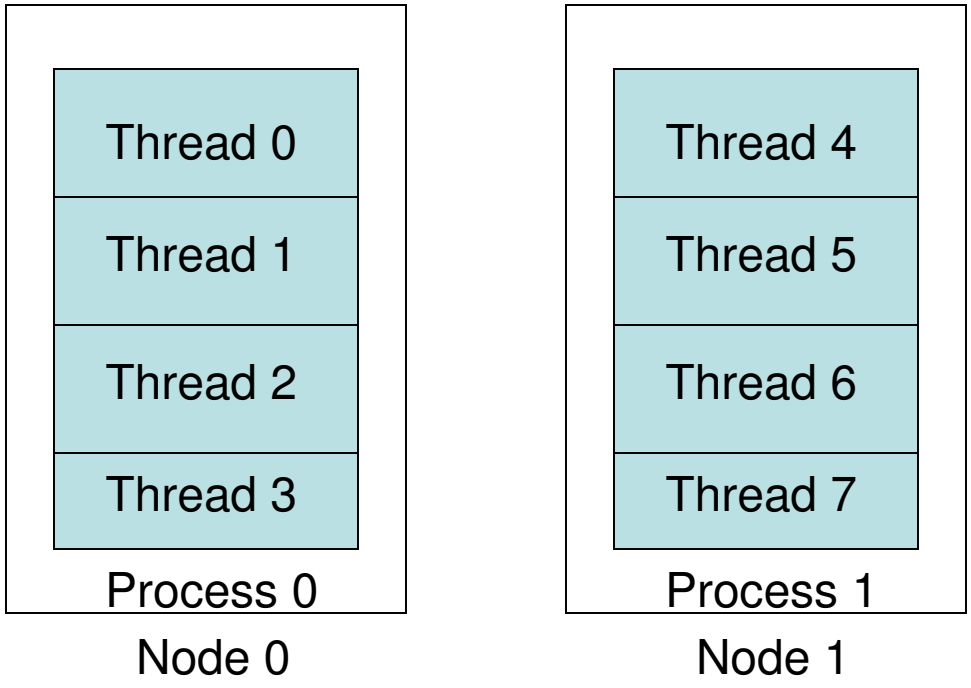
Основные принципы Cluster OpenMP

- После приложений минимальных усилий пользователя OpenMP-программа должна заработать на кластере.
- Самое трудное – найти параллелизм в задаче. Пользователь уже сделал это. Давайте извлечём максимум возможного из перенесённых страданий.
- Задаваемая OpenMP модель памяти позволяет эффективную работу не только на машинах с общей памятью: общие переменные констистентны не всегда, но в явно указанных местах.

Intel Cluster OpenMP: о продукте

- Специальный режим компиляции (-cluster-openmp) + доп. библиотеки.
- Распространяется вместе с компилятором начиная с версии 9.1.
- Для компиляции нужна отдельная лицензия, для запуска получившегося исполняемого файла – нет.
- Си/Фортран, Линукс, 64-разрядные платформы (Intel64 и совместимые, Itanium), TCP либо Infiniband/DAPL.
- Разработан на основе системы TreadMarks, лицензированной у Rice University.

Cluster OpenMP с точки зрения ОС

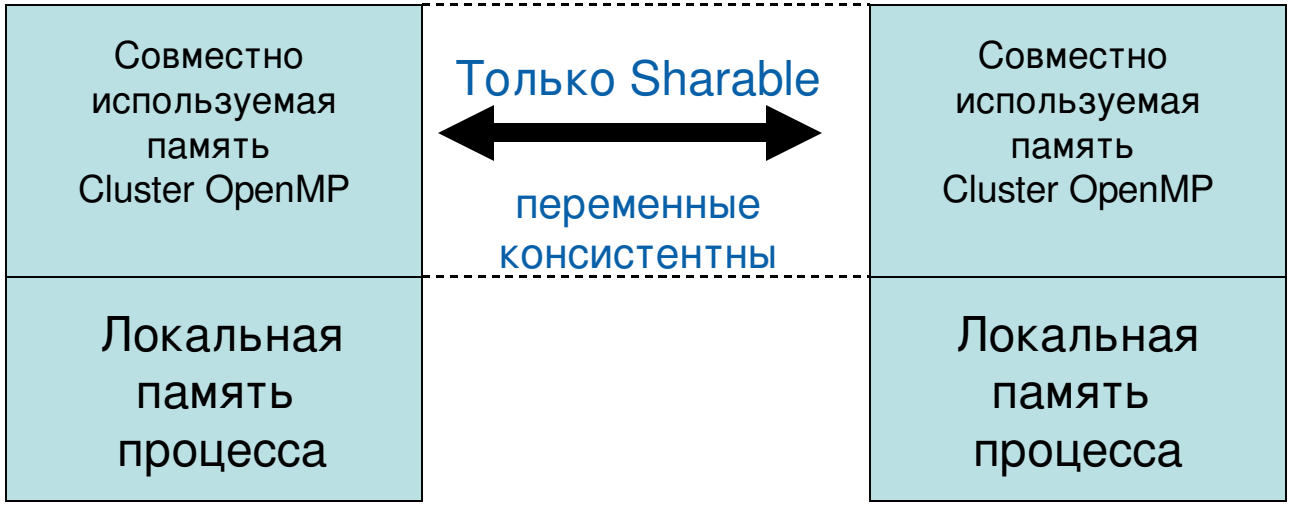


OpenMP нити разбрасываются по нескольким процессам, прозрачно для пользователя. Рабочий режим – по процессу на узел кластера, нитей на узле по числу процессоров. Для отладки можно пускать несколько процессов на одном узле.

Узлы для запуска, число процессов и число нитей в процессе указываются в настроечном файле. Есть возможность использовать механизм управления запуском процессов MPD из Intel MPI/MPICH2.

Cluster OpenMP: модель памяти

Если несколько OpenMP нитей обращаются к одной переменной, она должна быть sharable.



Cluster OpenMP Sharability

- Всё, явно или неявно описанное и используемое как shared должно должно быть описано как sharable. Объявлять системные переменные, например, файловые дескрипторы sharable недопустимо: Cluster OpenMP не обеспечивает представления системы в качестве единого образа. Например, открытые в разных процессах файлы – разные.

- Определяется автоматически компилятором, на основе опций компилятора, либо явным указанием.

- Для явного указания sharable используются директивы (это расширение OpenMP):

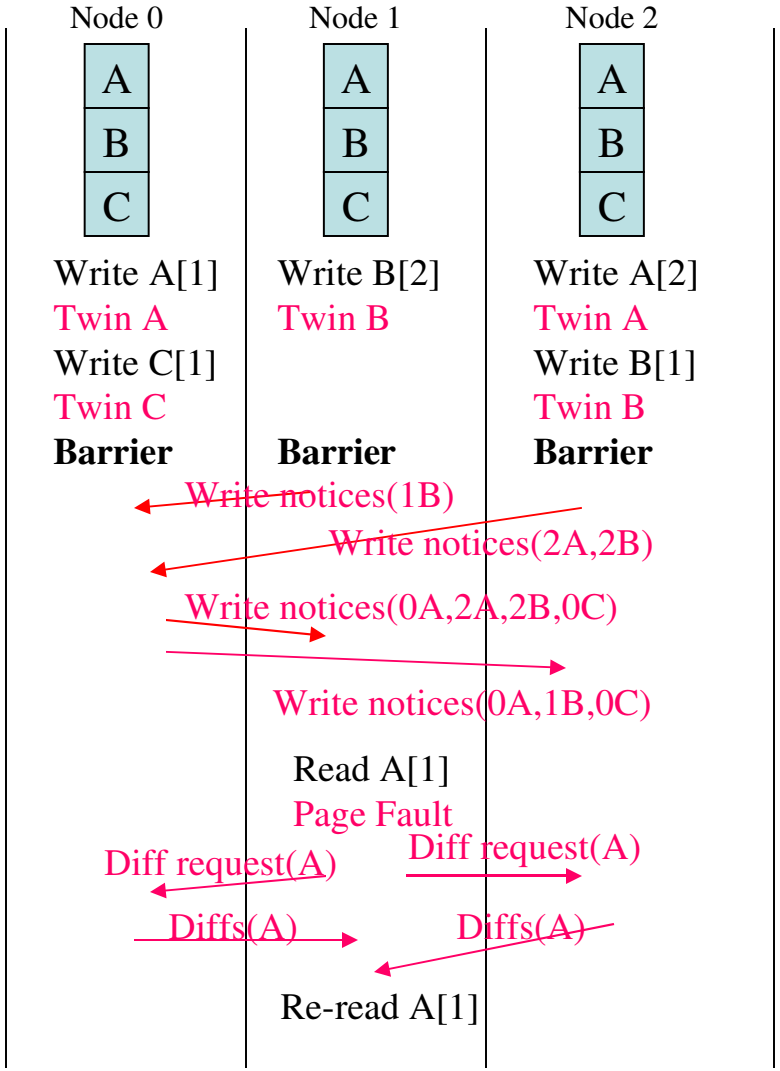
```
#pragma intel omp sharable(var)   // C,C++
!dir$ omp sharable(var)           ! Fortran
```

Как это работает: протокол обеспечения конситстентности, принципы

- Используется механизм защиты страниц и сигналы
- Когда принадлежащая к sharable памяти станица не валидна, меняем атрибут страницы
- По защите доступа вызывается SIGSEGV, который перехвачен CLOMP-библиотекой. Обработчик сигнала обращается к удалённому узлу за требуемыми данными.
- Защита со страницы снимается.
- Рестартуем прерванную команду, теперь уже ошибки защиты не будет.

Как это работает: протокол обеспечения консистентности, иллюстрация

1B означает, что узел 1 пишет в станицу B.



Инструменты отладки эффективности

- Intel Trace Analyzer: хорошо знакомые цветные полоски, позволяющие увидеть кто, когда, чем и с кем менялся.
- segvprof.pl. Собирается статистика о ошибках защиты страниц, привязанная к исходным текстам.
- Intel Thread Profiler. Что происходило на уровне одного процесса.

Превращение OpenMP-программы в CLOMP-программу

- Главное, что нужно добавить – сделать sharable те данные, которые используются как sharable.
- В некоторых случаях для поиска может использоваться:
  - компилятор (см. clomp-sharable-propagation)
  - run-time (см. KMP_DISJOINT_HEAPSIZE).
- 3 способа сделать данные sharable
  - #pragma intel omp sharable / !dir$ omp sharable
  - ключи компилятора (напр., сделать все common-блоки sharable)
  - выделение динамической памяти в sharable области.