Subject: Re: Vector performance on a specific situation
Posted by Novo on Tue, 18 Jun 2013 17:11:09 GMT
View Forum Message <> Reply to Message

crydev wrote on Tue, 18 June 2013 03:01Hello,

I have a question about the Vector's performance in a specific situation. I have a program that utilizes 8 threads on new systems, heavy utilization of paralellism. Say I have a Vector containing 300 items. I split the indexes of those items over 8 threads, meaning the Vector will be accessed from 8 threads simultaniously, but every thread accesses a different item. The same memory location is never modified.

I have read something about Vector cache lines. What is the performance of the U++ implementation of the Vector in this situation? I tried to copy the thread-specific data into arrays and passed them into the functions, but it seems like just as fast.

If there is a better way to do this, I appreciate any suggestions.
If you are just reading data there will be no problems. But if you write to elements (even if they are not shared among threads) you get yourself into false sharing problem. Basically, the idea is that CPU doesn't work with words, it works with cache lines. The simplest way to fix that is to add padding to your data. Example: instead of using raw int you can use a structure below.

```
struct MyData {
    int data;
    char padding[64 - sizeof(data)];
};
```

Size of cache line is usually 64 bytes, so you need to add padding to make you data land onto different cache lines.