Subject: Re: cpu times
Posted by Gianluigi Boca on Thu, 31 May 2012 19:20:27 GMT
View Forum Message <> Reply to Message

dear Felix,
I repeat here my reply since I don't see it in the discussion
and I fear it may have been lost.

So, the reason why I considered the at() function was beacause
in my opinion it is the more useful functionality of <vector>
(the boundary check).

Anyway I have measured also the [] operator and it turns
out to be 'only' about 4.5 times slower than the traditional
C array access (on a 64-bit Lenny machine here at GSI).

That in principle is still way too much, I believe.

However, after a discussion with Mohammad, he had promised to assess with Valgrind how much time is spent in the Pandaroot code on average in the STL library compared to the total process
time, for some 'typical' Panda event reconstruction etc.

If that fraction of time is negligible he says it is worthless
to bother.

Some computer gurus may disagree with his point (every fraction
of Cputime saved may translate in the long run in many days of Cputime saved). I have already heard this discussion in the past.
But anyway, let's stay tuned and see what he founds

cheers Gianluigi

Felix Boehmer wrote on Sat, 12 May 2012 11:14Dear Gianluigi,

while these are interesting measurements, I have to re-cite one those fun meetings we had last summer: You are comparing apples with pears!

If you want to compare the raw performance of the two data classes, you have to use similar functionality, e.g. the [] operator of the <vector> which does no implicit range check. It is unnecessary and bad practice to use at() in loops of the kind for(int i=0; i<vector.size(); i++) { meh = vector.at(i); //Completely unnecessary range-check
}over vectors anyway.

It would be interesting to directly compare assignment and reading performance like you did by replacing at() with []. Another thing you could look at which would have some real-world relevance is to compare array[] and (*<vector-pointer>)[] performance, that is the combined performance of a necessary de-referencing with following raw access.

Cheers