LONDON2007

P214

Efficient Computation of Passive Seismic Interferometry

J.W. Thorbecke* (Delft University of Technology) & G.G. Drijkoningen (Delft University of Technology)

SUMMARY

Seismic interferometry is from a computationally point of view based on cross-correlating two signals. One application of seismic interferometry lies in passive seismics. In passive seismic recorders are placed on the surface of the earth and signals are continuously measured. In this abstract we investigated an efficient way of computing the cross-correlation, of a limited number of output samples, for continuously measured signals. Efficiency tests are carried out on two types of common of the shelf 64-bit CPU's: Intel Xeon and AMD Opteron.

Introduction

We have a number of passive measurement stations which are continuously recording (with for example 1 ms sampling in 24 bit) signals from the surface and subsurface of the earth. These recordings are being correlated with each-other to simulate seismic reflection data. In this abstract we mainly concentrate on an efficient implementation of the correlation kernel and how to deal with the amount of data.

For a good response after the cross-correlation a very long recording time and many spatially uncorrelated white-noise sources are needed. The longer the recorded response, the better it approximates diffuse fields. At the same time, with longer recording times the response from more noise sources is recorded, and as a result, one obtains better illumination of the subsurface. The theory of seismic interferometry has gained a lot of attention in the past 5 years and is for example derived and explained in Draganov *et al.* (2006); Wapenaar and Fokkema (2006). The basic correlation relation, which is of our interest, is given by:

$$R(\mathbf{x}_A, \mathbf{x}_B, t) + R(\mathbf{x}_A, \mathbf{x}_B, -t) = \delta(\mathbf{x}_{H,B} - \mathbf{x}_{H,A})\delta(t) - T_{obs}(\mathbf{x}_A, t)T_{obs}(\mathbf{x}_B, -t) \quad (1)$$

where $\mathbf{x}_H$ denotes the horizontal coordinates (x_1, x_2). Here $R(\mathbf{x}_A, \mathbf{x}_B, t)$ is the reflection response of an inhomogeneous medium in $\mathcal{D}$, including all internal multiples, for a source at $\mathbf{x}_B = (\mathbf{x}_{H,B}, x_{3,0})$ and a receiver at $\mathbf{x}_A = (\mathbf{x}_{H,A}, x_{3,0})$. $T_{obs}(\mathbf{x}_A, t)$ is the transmission response of uncorrelated (noise) sources in the subsurface, including all free surface multiples, with receivers at $\mathbf{x}_A$ on the surface of the earth. Equation (1) shows that the correlation of transmitted signals measured at different position at the surface of the earth gives reflection data. This reflection data is one of the main goals in passive seismics.

The measured 'noise' signal

The measured signals $a_i(t)$ at X_i are assumed to contain transmission events from small earthquakes (caused by layer slides, reservoir depletion) in the subsurface. Besides these transmission events, from deeper parts in the subsurface, the signals will also contain a lot of surface waves caused by 'disturbances' at the surface (traffic, sheeps, people, wind, ...). Our main interest are the transmission events from the deep subsurface. The initiation time of the transmission events is unknown, we assume/hope this will happen a few times a day. By summing over long intervals different events are added together and will enhance the constructed reflection series. We assume that the response of the earth of the top layers we are interested in does not change during the (long) time of the measurements. The events we are interested in will always occur in a short time window (e.g. 10 s.). In Figure (1) a signal with the transmission events occurring at different times t_e^i is shown.

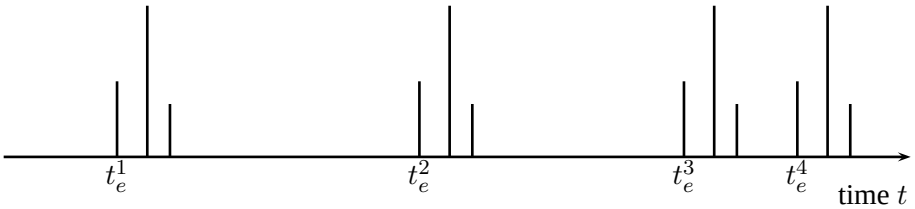

Figure 1: *The simplified measured signal contains transmission events which occurs at unpredictable times t_e^i. Each event contains a series of back-reflections from the surface of the earth, and internal reflections between layers within the subsurface.*

Continuous on-line correlation

Given the two real signals $a(t)$ and $b(t)$, we can compute the cross-correlation as:

$$c_{ab}(t) = a(t) * b(-t) = \int_{-\infty}^{\infty} a(\tau)b(\tau + t)\, d\tau \qquad \forall \quad t \in (0, T_c) \tag{2}$$

where the correlated signal has a length T_c. This length must be long enough to contain the reflections we are interested in, e.g. 10 seconds. This length is bound by the physical position of the reflectors. Note that cross-correlation is not commutative and $c_{ab} \neq c_{ba}$.

The length of the original input data to be correlated is infinitely long, expressed by the bounds of the integral above. In practice we will have to start somewhere, say at time 0, and we can only correlate up to a certain large time, which we call T_α. Then we have:

$$c_{ab}(t) = \int_{0}^{T_\alpha} a(\tau)b(\tau + t)\, d\tau \qquad \forall \quad t \in (0, T_c) \tag{3}$$

Due to the end points of the total correlation window it is clear that we make an error here.

Our aim is to design a correlation algorithm which continuously computes the correlation result. In this way we can avoid the storage of very long signals. This approach is possible because we are only interested in a short output time-window (T_c). Using the limited number of output samples, the integral above can be written as a finite sum of smaller windows, i.e.:

$$c_{ab}(t) = \sum_{k=0}^{N_w-1} \int_{kT_k}^{(k+1)T_k} a(\tau)b(\tau + t)\, d\tau \qquad \forall \quad t \in (0, T_c) \tag{4}$$

where N_w is the number of windows and T_k is the smaller window in which a correlation is calculated.

The discrete version of equation (4) is

$$c_{ab}[i\Delta t] = \sum_{k=0}^{N_w-1} \sum_{j=kN_k}^{(k+1)N_k-1} a[j\Delta t]b[(j + i)\Delta t] \qquad i \in [0, N_c - 1] \tag{5}$$

where N_k is the number of samples in the window T_k and N_c is the number of time samples of the output signal, i.e., in the window T_c. Equation (5) shows that, using smaller N_k windows, N_c output samples can be correlated as accurate as equation 3.

The number of windows times the number of samples in each window should be equal to the number of samples in the window from 0 to T_α, which we call N_α. We then have:

$$N_w N_k = N_\alpha \tag{6}$$

Equation (5) makes an efficient data processing scheme possible. In this scheme data is buffered for a short period (T_k), as can be seen in equation (5). The buffered data is used to compute the correlation. During the compute time of the correlation new data is buffered.

Note that we have defined four N's, so four lengths:

1. N_α used for the number of samples in total correlation window $(0, T_\alpha)$;

2. N_w used for the total number of smaller windows in the correlation;

3. N_k used for the number of samples in the small correlation window with length T_k;

4. N_c used for the number of samples after the correlation, i.e., of the output signal.

Choice of N_k: We know that the number of output points N_c is limited, so a convenient choice for N_k is: $N_k = 2 * N_c$. For this choice we can generate N_c output points with the buffered signals. After the computation of the first N_c samples the buffers can be refilled with N_c new samples.
Two straightforward numerical implementations have been made, one which uses copies and one which uses pointer references, to use the data in the buffered arrays. Of course, it is expected that the pointer references will be the faster way, given that the used compiler can generate efficient code with pointer references.

CPU efficient correlation

The correlation has to be implemented efficiently on hardware to exploit the specific features of the hardware at best. We tested the efficiency on two types of CPU's, namely on the 64-bit Opteron of AMD, and on the 64-bit Xeon of Intel. The main difference between the CPU's are the caches and the connection of these caches to the RAM. On the Intel system, a North Bridge links the two, while in the AMD-system an on die memory controller is used.
A simple correlation kernel has been used to determine the optimal block size (nt) for the CPU.

```
for (l=0; l<ntout; l++) {
    for (j=0; j<nt; j++) {
        c[l] += a[j]*b[j+l];
    }
}
```

This kernel is the implementation of equation (5), where $N_w = 1$, `ntout`=N_c and `nt`=N_k. The correlation code needs to load for every iteration three 4 Byte float (assuming load-store architecture) and uses an add. and a mult. per compute step. This means that the code is limited by the memory bandwidth.

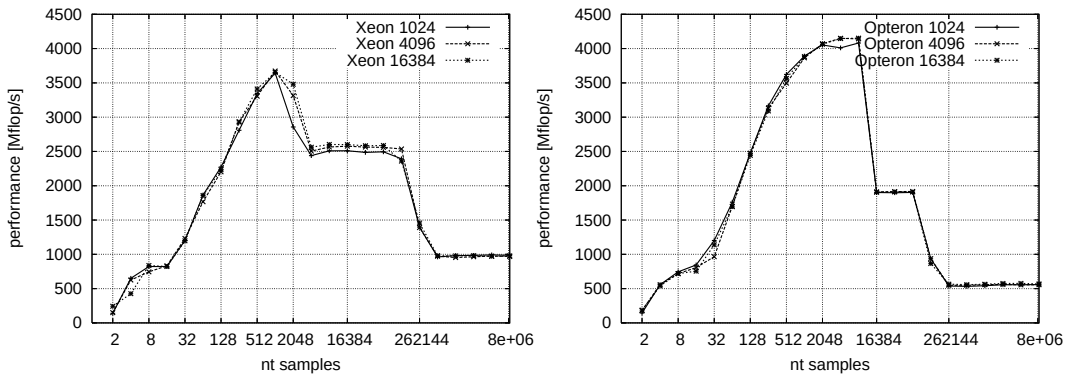

Figure 2: *Computational efficiency of a correlation kernel. Left the performance (in Mflop/s) results for a Xeon 3.0 GHz 16K L1-, 2 MB L2-cache: and right for an Opteron 2.2 GHz 64K L1-, 1 MB L2-cache.*

The results of this simple kernel, with varying `ntout` and `nt`, are shown in Figure 2. To compute the code the Intel compiler (version 9.0 with flags -O3 -axP) has been used. The code has been run for three different values of the `ntout` parameter: 1024, 4096 and 16384.

Let us first focus on the Xeon chip from Intel. Assuming the code can load one 4 Byte float per cycle (1/(3 GHz) s.) from the L1-cache, then the maximum peak for this code on the Xeon is 2/3*peak performance = 4.0 Gflop/s. In the left picture of Figure 2 the maximum observed peak is 3.6 Gflop/s (the difference with the computed maximum peak of 4.0 Gflop/s is unclear). In the figure we see a first drop in performance above 1024 samples. This is due to the L1-cache whose size is 16K and can contain 3x1365 4 Byte floats. The next drop in performance is from L2 (2 MB) to main memory, occurring above 131072 samples. For the correlation we should aim at block-sizes between the L1 and L2 cache, in this case 1024 samples would be the best choice.

Let us next focus on the Opteron 248 (2.2GHz) from AMD, as shown in the right hand-side picture in Figure (2). The Opteron has a similar L1/L2 cache structure as the Xeon, but the connection to main memory is different. Where the Xeon has to go to the NorthBridge to go to main memory, the Opteron uses an on-die memory controller. Two HyperTransport ports to the DDR 400 memory dimms gives a bandwidth of 6.4 GB/s. The L1 cache of the Opteron is 64K and the L2 cache is 1 MB. As can be seen on the figure, the performance reached (above 4.0 Gflop/s) is not only higher than the Xeon, but it also reached at larger block size, namely at 8192 samples. The next drop drom the L2 cache to the main memory occurs above 65536, so earlier than for the Xeon. From this graph it can be seen that the correlation runs optimally at 8192 samples.

To make optimal use of the compute hardware the correlation loop should be blocked on the size of the L1/2 cache of the used CPU.

Conclusion

For continuously measured signals the correlation, with a limited number of output samples, can be computed during the recording of new data. Buffers with a limited number of time samples are used to store incoming data during the time the correlation is computed. The correlation result can also be used as an efficient way of data reduction.

Acknowledgment

We would like to acknowledge Advanced Micro Devices (AMD) for giving us an Opteron based server.

References

Draganov, D., Wapenaar, C.P.A., and Thorbecke, J. [2006] Seismic interferometry: Reconstructing the earth's reflection response. *Geophysics* 71(4), SI61–SI70.

Wapenaar, Kees, and Fokkema, Jacob. [2006] Green's function representations for seismic interferometry. *Geophysics* 71(4), SI33–SI46.