S05: High Performance Computing with CUDA

Case Study: Wave propagation through earth

Bernard Deschizeaux
CGGVeritas Oil&Gas service company

Outline

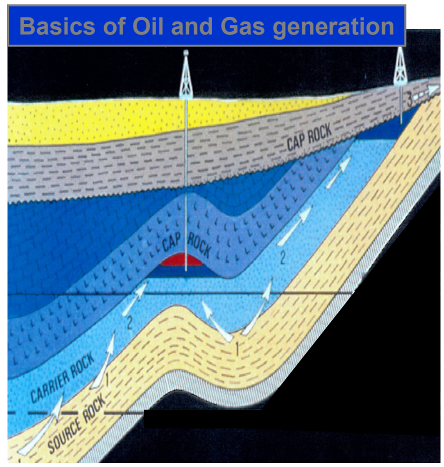
- What Oil & Gas industry look for
  - Introduction to oil prospection.
  - Seismic prospecting overview.
- What CGGVeritas offers
  - Data acquisition.
  - Data processing.
- Overview of main processing steps
  - Description of the problem to solve.
  - How do we solve it.
- Converting the algorithm in CUDA
  - Kernel example.
- Industrial environment in processing datacenter
  - Using CUDA in hybrid parallel environment.
  - Monitoring tools.
  - GPU server solution.
- Conclusion

What Oil & Gas look for

CGGVERITAS

- Oil is generated deep down the earth.
- Oil migrate up through porous rocks.
- Oil is trapped by none porous rocks with specific structural shapes.

- To find oil the knowledge of the earth structure and rocks property is mandatory.



Economical pressure for accuracy

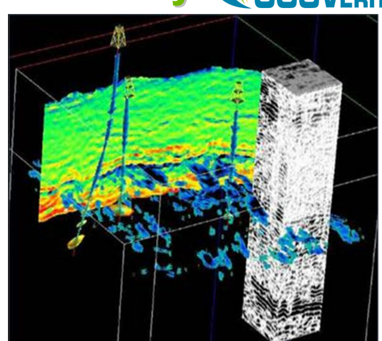
CGGVERITAS

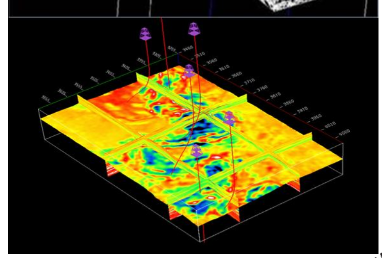
- Mature fields need from 10 to 100 wells.

- Typical cost for drilling a well is 10 millions dollars.

- At actual stat of the art 20% of the drilled wells are lost!

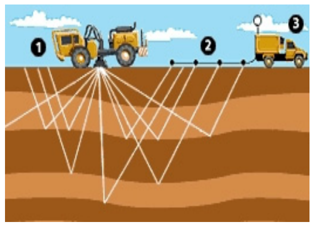
- Wells need to be positioned at a few meters error, below 1km of sea water and few km of rocks.

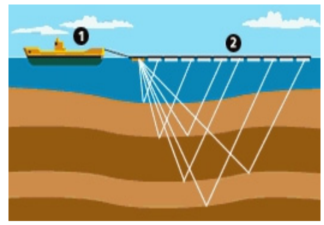




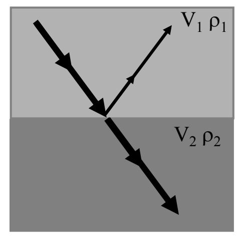
Earth studies: Seismic survey

- Wave propagation in the earth is related to rocks properties



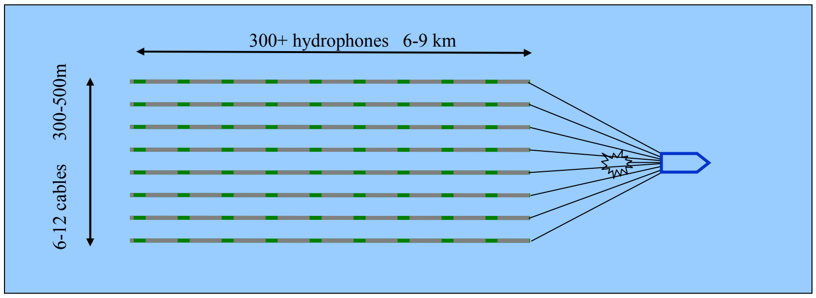


- Arrival time of reflection gives structural information.
- Amplitude of reflection gives information on rocks properties. (wave velocity, rock density)



CGGVeritas Today

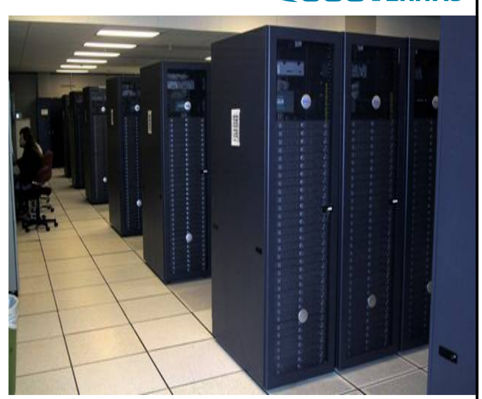
1 seismic survey = 10TB of data



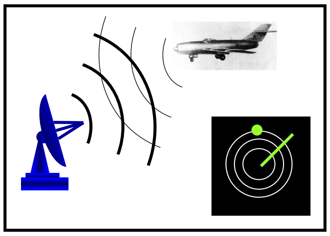
- *One shot ~ 5000 records of 10KB.*
- *Typical accuisition:10^5 shots to sample the earth every 25m*
- *10TB of raw data.*

Seismic processing

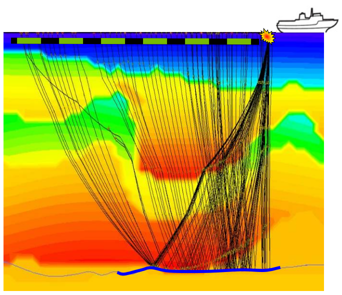
- *10TB of raw data.*
- *Every byte of data will need 10^6 operations.*
- *Through the processing, data size will be increased tenfold.*
- CGGV processing capability is centralized in main datacenters providing hundreds of Teraflops of computer capability and PetaBytes of storage.



Converting arrival time into depth

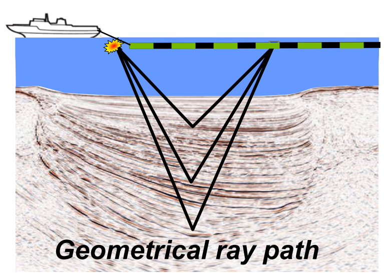


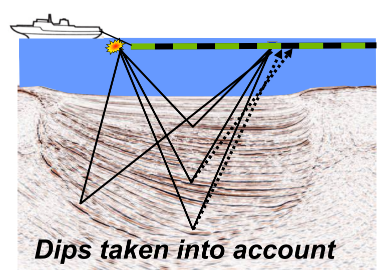
- In case of radar distances are simple to extract from arrival time: D = T* V/2.
  - Constant velocity in air.
  - Straight line propagation.

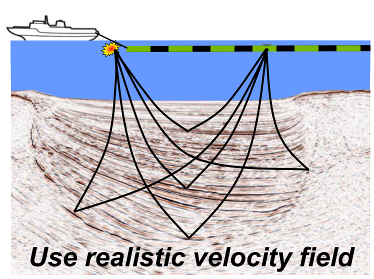


- In Seismic exploration the problem is far more complex.
  - Far more complex reflections.
  - Velocity field is unknown.
  - Velocity field is inhomogeneous.
  - ⇒ Ray path are not straight.

Migration: dips & wave velocity effect



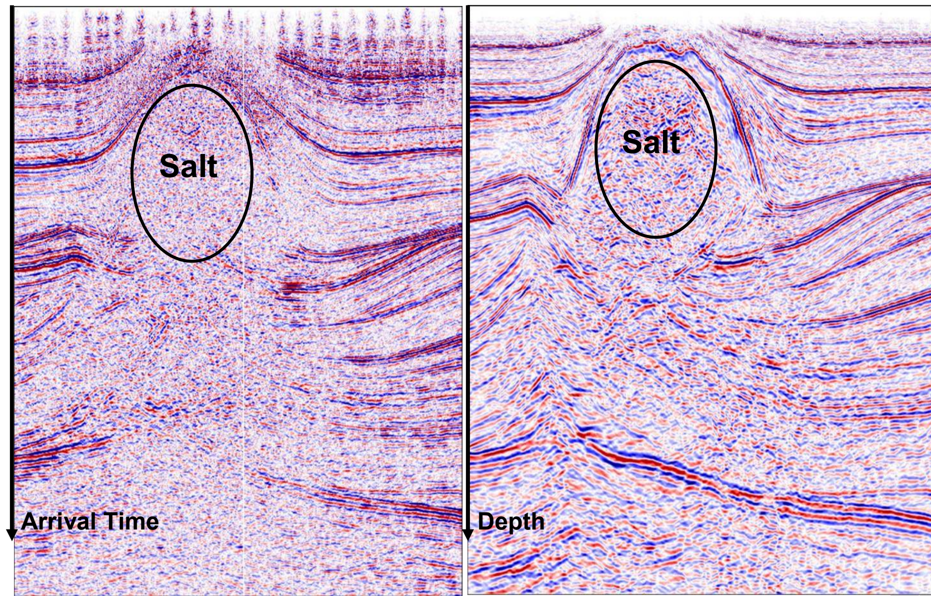




- *For correct image we need the velocity field.*
- *The wave needs to be propagated in all directions.*

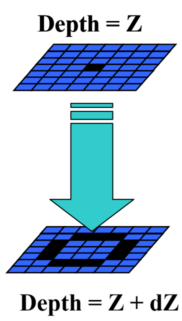
Migration results example

CGGVERITAS



The wave propagation algorithm

CGGVERITAS



- **Finite difference algorithm used in frequency to propagation from one depth to the next.**
- **Propagator expressed in the spatial direction using Chebychev polynomials of second derivative.**

$$G(\Delta) = \sum_{n=0}^{n=N} a_{\omega/V}(n) T_n(\Delta)$$

- **T_n(x): Chebychev polynomial of degree n computed using the recursive equation:**

$$T_n(x) = 2x.T_{n-1}(x) - T_{n-2}(x)$$

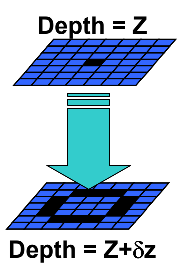
- **Δ the second derivative is approximated by a 2D cross filter.**

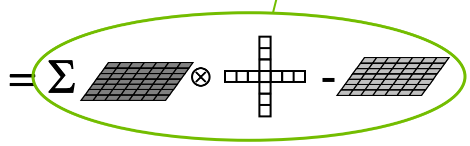
Reference: Soubaras, R., 1996. “Explicit 3-D migration using equiripple polynomial expansion and Laplacian synthesis”. Geophysics, Volume 61, pp. 1386_1393

Propagator algorithm

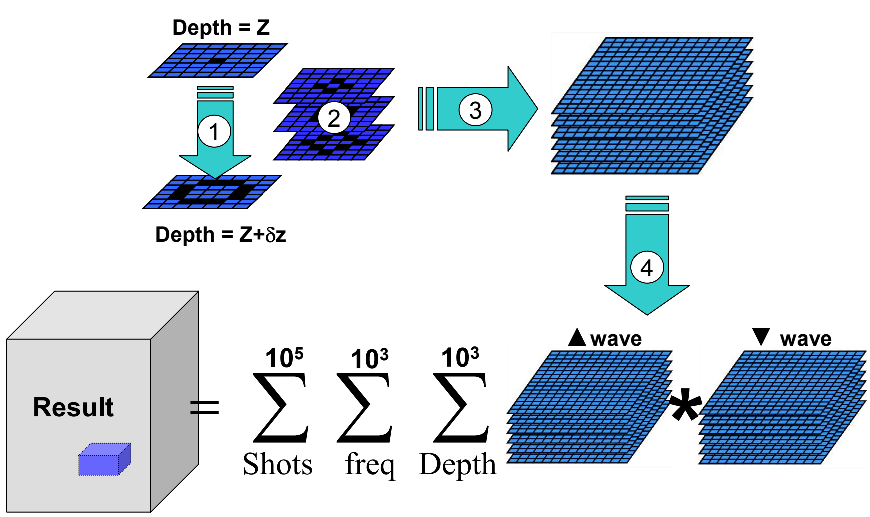
```
T(x,y) = Wave_z(x,y);
TT(x,y)=Δ⊗Wave_z(x,y);
Wave_zdz(x,y) = α_1*T(x,y)
              + α_2*TT(x,y);

for (n = 2; n < NMAX; n++){
   // Compute the Chebychev polynomial TTT
   TTT(x,y)=2*Δ⊗TT(x,y)–T(x,y);
   // Store Chebychev results for next iteration.
   T(x,y) = TT(x,y);
   TT(x,y)= TTT(x,y);
 // update result.
   Wave_zdz(x,y) += α_n*TTT(x,y);
};
```





10^{11} propagations for the final image

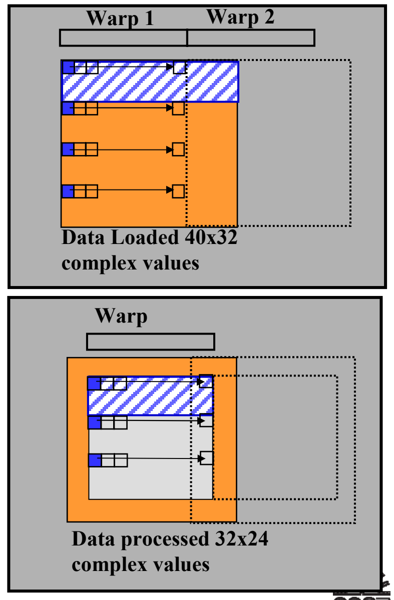


CUDA propagator kernel

Block geometry: 64x8 = 512 threads

- Load:
  - Each thread reads one value in each quarter of the block data.
- Processing:
  - Each thread applies the cross operator to one value in each third of the block data.

Random access for the kernel.
But sequential access for all warps of a block.



Complex performance analyses

- Kernels speedup varying from none to one order of magnitude.
- More than 100 parameters.
  - Large varieties of survey geometry.
  - Input frequency plan size from 50x50 to 500x500.
  - Pre and Post processing percentage (Amdahl law).
  - Interpolation increment, operator length
- Global speedup even more difficult to extract.
  - Finding representative dataset.
  - Finding representative parameters (empirical).
  - Knowing that even 2X speedup will change drastically the whole methodology.

CUDA development

- Development is not straightforward.
- Programming model very powerful and flexible.
- Lack of debugger and profiler.
- Some weakness in error handling.
- CUDA still in the evolution curve (more functionality and less constraint yet to come).
- Preliminary performance results on individual kernels very encouraging.
- Ongoing benchmark on different datasets and with realistic parameters.
- Hardware GPU server solution still in evaluation.

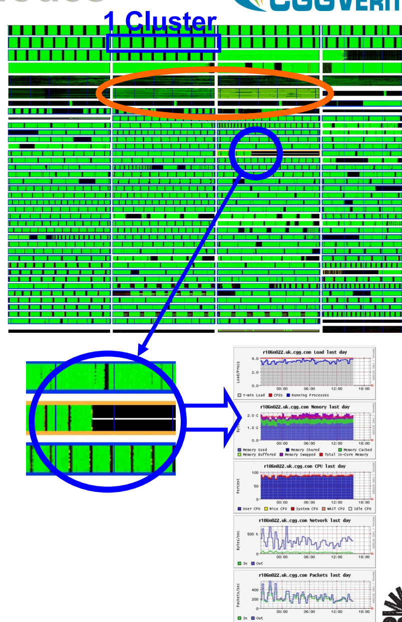
Development for processing hub

The prototype is included in a composite multi-layers parallelism

- Many processing project running concurrently.
- Large processing step split and schedule on different PC cluster racks.
- MPI used to parallelize on nodes in a rack.
- OpenMP used to parallelize on cores in a node and to schedule multi GPU.
- CUDA used to parallelize on GPU.

Monitoring 1000th of Nodes

- For a processing hub of many thousands of nodes, hardware failures occur every day.
- Monitoring tools are needed to detect and analyze problems and efficiency.
- We cannot afford to lose more than a few hours processing because of failure.
- Failure rate of every piece of hardware needed to scale job size and backup save rate.



GPU in a processing datacenter

- Performance measurement
  - Speed Up.
  - Overall percentage of applications using GPU.
- Large datacenter constraints.
  - Space.
  - Power supply.
  - Cooling system.
  - Maintenance.
  - Price.
- What GPU solution need to provide
  - Compact (1u server).
  - Reasonable power needs.
  - Integrated in the general cooling system.
  - Stable for sustain use.
  - Inexpensive.

Conclusion

- Collaboration with NVIDIA is fruitful.
- Industrial time consuming algorithm successfully prototype with CUDA.
- Extensive benchmark ongoing on real scale datasets.
- Design for GPU server still in progress.
- Capability trend of GPU versus CPU show that this is a promising technology for number crunching solutions.