

Estimating the Performance of a GEDAE Graph

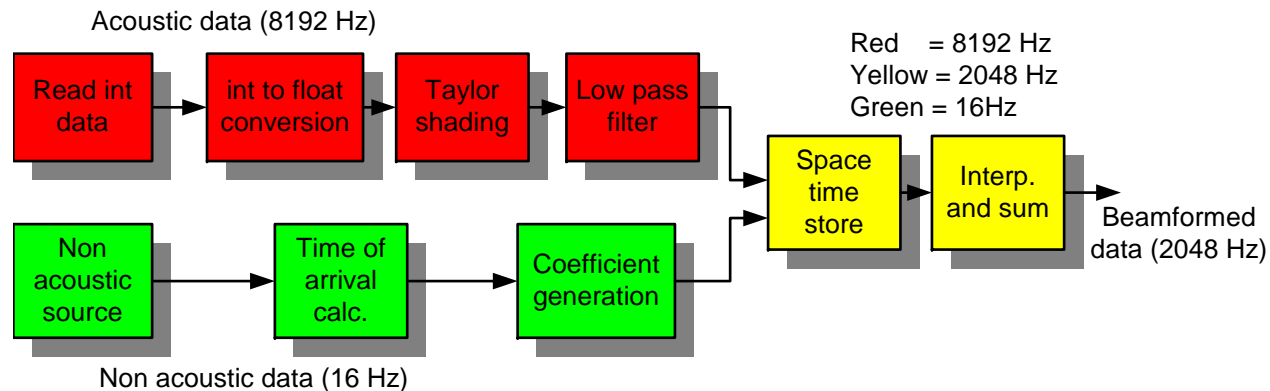


Simon Challands

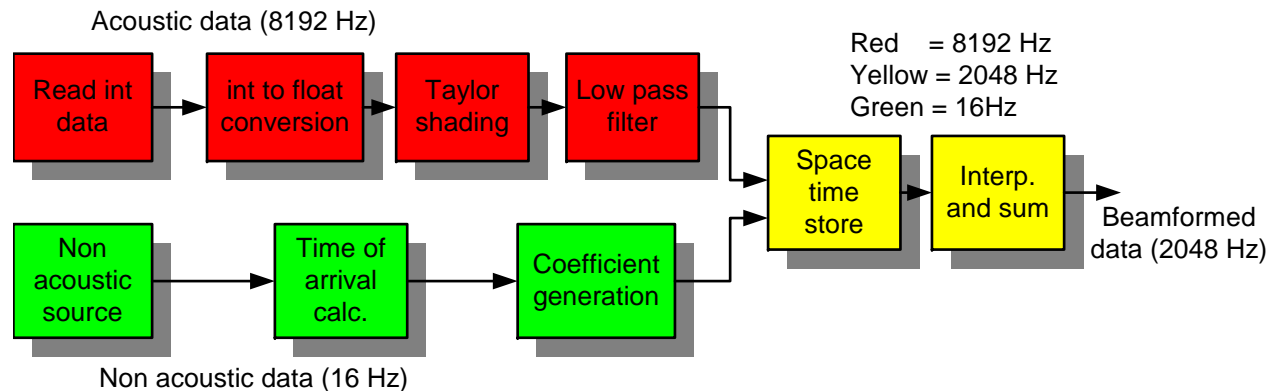
What are the computational overheads of GEDAE?

- Primitive box code
- GEDAE's scheduler
- Gathering trace table measurements
- Effects of granularity on memory efficiency (e.g. page faults)

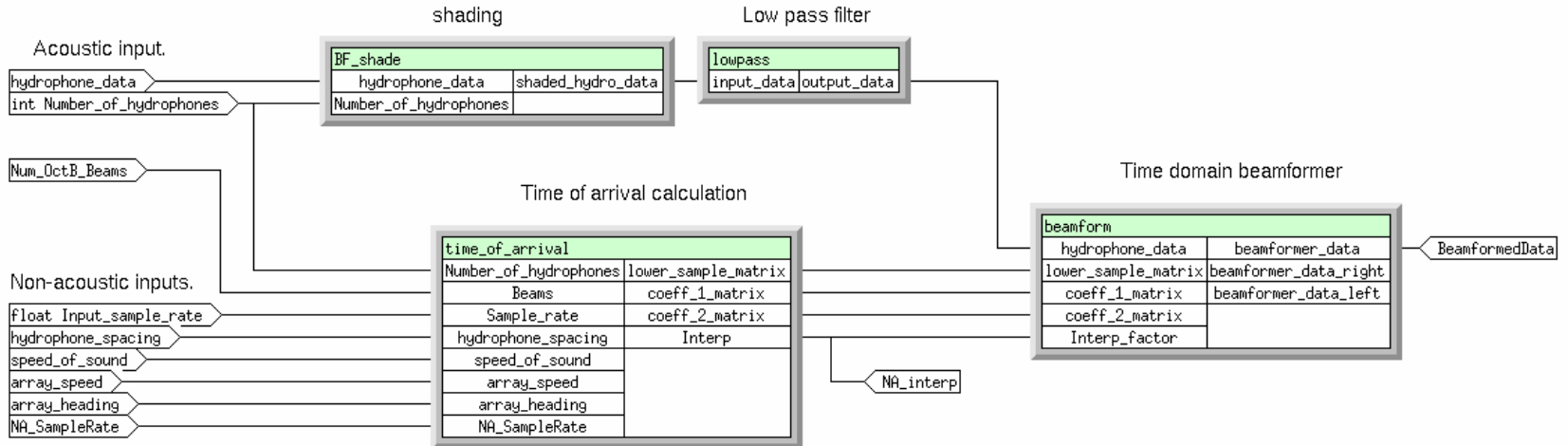
- Beamformer is a series of predictable operations
- Dataflow is straightforward, so the number of operations is predictable
- Data rates chosen for convenience of calculations, and are not representative of any particular system



- The number of discrete operations in each box (e.g. multiply-accumulates) can be estimated
- Average times for these operations are known
- Difference between these estimates and measured time for each box is the estimate of the GEDAE overhead



- The GEDAE graph consists of both custom-written and built-in primitives



- **Trace table, on each box**
 - Each box can be timed separately
 - Can compare each component against predicted load
 - Gathering trace statistics adds its own overhead

- **Trace table, using the lines**
 - Gives the complete performance
 - No individual boxes

- **Self-timing code**
 - Only directly affects one box
 - No additional trace table overheads



Operation	Actual operation	Cost	Fir	Npts	Beams	Staves	F.Calc	Mop/s	Operat	Pentium 2.8GHz	total	+marg
										Mcy/s		
							(Hz)			Theor		
Enet load in					64	8192		0.524	io	25.17		
Float	Si to S				64	8192		0.524	fixflot	2.10		
Aperture weighting					32	8192		0.262	macr	1.05		
Low pass filter				111	32	8192		29.1	macr	116.39		
Coeff gen	Time of arrival	3			64	32	16	0.098	macr	0.39		
Coeff addressing	Shift and coeff	6			64	32	16	0.197	macr	0.79		
Data Accessing	Vspace time store		2		64	32	2048	8.389	rwExt	58.72		
2pt BFM	Interpolation & sum		4		64	32	2048	16.78	macr	67.11		
Enet load out					64		2048	0.131	io	6.29	278	333.6

Operation	Frequency of operation	Amount of data processed	Predicted time (10 ⁶ cycles / s)	Measured time (10 ⁶ cycles / s)	Difference
Data in*	8192 Hz	64 hphones	25.17	19.23	76%
int to float	8192 Hz	64 hphones	2.10	3.89	185%
Weighting	8192 Hz	32 hphones	1.05	3.72	355%
Low pass filter	8192 Hz	32 hphones, 111 coeffs.	116.39	130.40	112%
Time of arrival	16 Hz	64 beams, 32 hphones	0.39	0.36	92%
Shift and coeff. calc.	16 Hz	64 beams, 32 hphones	0.79	2.07	263%
Spacetime store	2048 Hz	64 beams, 32 hphones	58.72	66.52	113%
Interpolation and sum	2048 Hz	64 beams, 32 hphones	67.11	150.87	225%
Others	-	-	54.34 (20% error estimate) Plus unknown OS overhead	51.28 (upsamp. etc)	113%
TOTAL			326.06	428.34	131.37%

* Load is actually network read in predicted, and disc in measured

- **Some boxes much more efficient than others**
 - **Weighting box is very slow (355% of predicted loading)**
 - **Simple GEDAE primitive (v_multVK)**
 - **Lowpass filter box is close to predicted (112% of predicted loading)**
 - **Hand-optimised box**

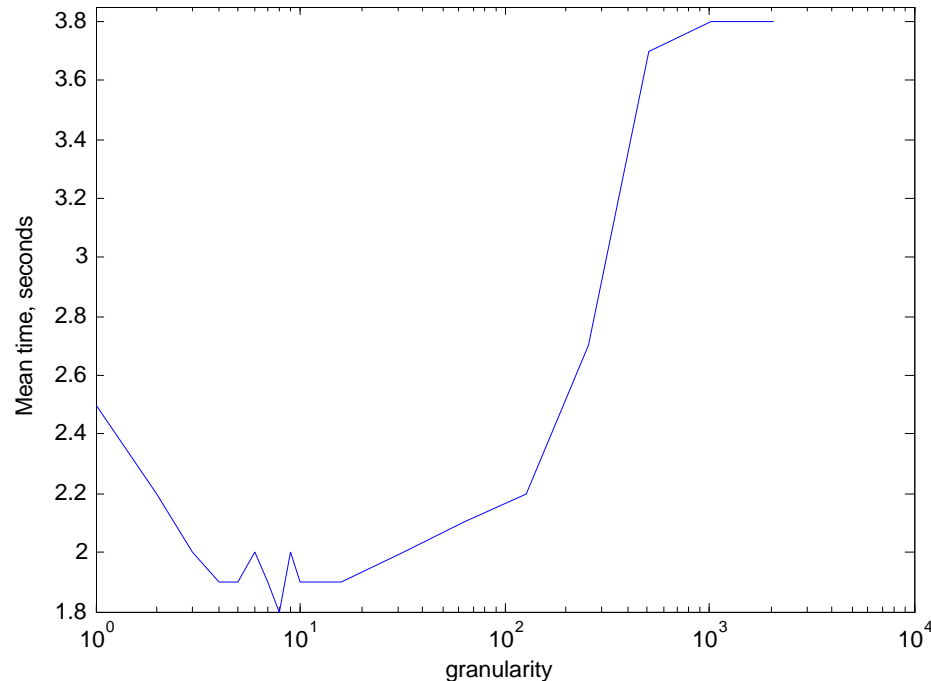
- **v_multVK apply method**

```
Apply: {  
    int N = size(in)/n;  
  
    while (N--) {  
        e_vmul(in,1,VK,1,in,1,n);  
        in+=n;  
    }  
}
```

- **size, while loop, and function interface are additional overheads from the apply method every time the box fires**
- **n = 32, for larger n more time would be spent in the e_vmul function, so relatively less in the overheads**

- **Hand-optimised version of v_efird**
 - **No function call**
 - **Loop unrolled**
 - **The box is therefore spending most of its time processing**
 - **Original GEDAE box achieved 332 Mcyc / s**
 - **Modified version achieves 130 Mcyc / s**

- **Time taken to process 10 second's worth of data**
 - As granularity increases the amount of memory required by each box increase
 - Operating system is therefore more likely to have to page memory in and out
 - Box's data less likely to fit in processor cache



- Trace table has to spend time gathering statistics
- Trace table is taking 4.3% of the time in this example
- Will be dependent on the number and complexity of primitives
- Example without trace table is therefore 125% of the predicted loading

Mode	Time to process 10 seconds of data	Percentage processor loading
With trace table	1.61 seconds	16.1% (450.8 Mcyc/s)
Without trace table	1.54 seconds	15.4% (431.2 Mcyc/s)

- **GEDAE graph efficiency can be made comparable to handwritten code**
- **Example graph runs at 125% of the predicted load**
- **Overheads for anything other than the simplest boxes are small**
- **Large collections of simple boxes will probably be less efficient than a single primitive doing the same job**
- **Effect of granularity changes cannot be predicted easily**
 - **Need to experiment to find optimum granularity**