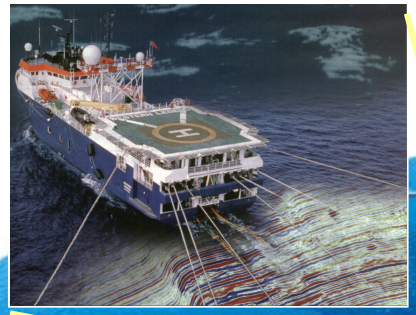
Processamento Sísmico de Alto Desempenho na Petrobras

Thiago Teixeira

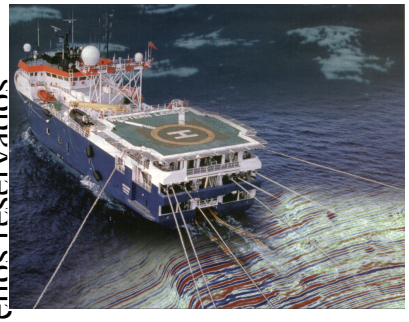
E&P-Exp/Geof/Tecnologia Geofísica

Julho/2011

Processamento Sísmico e Interpretação

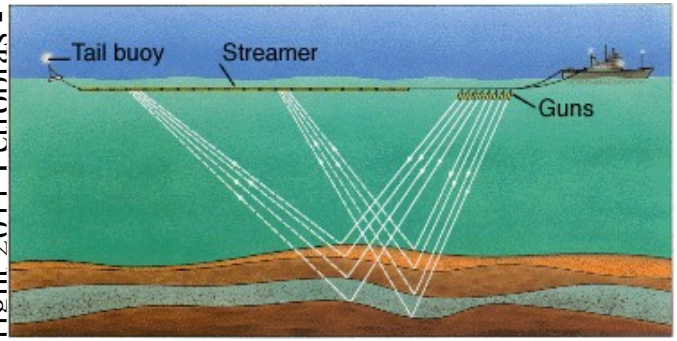


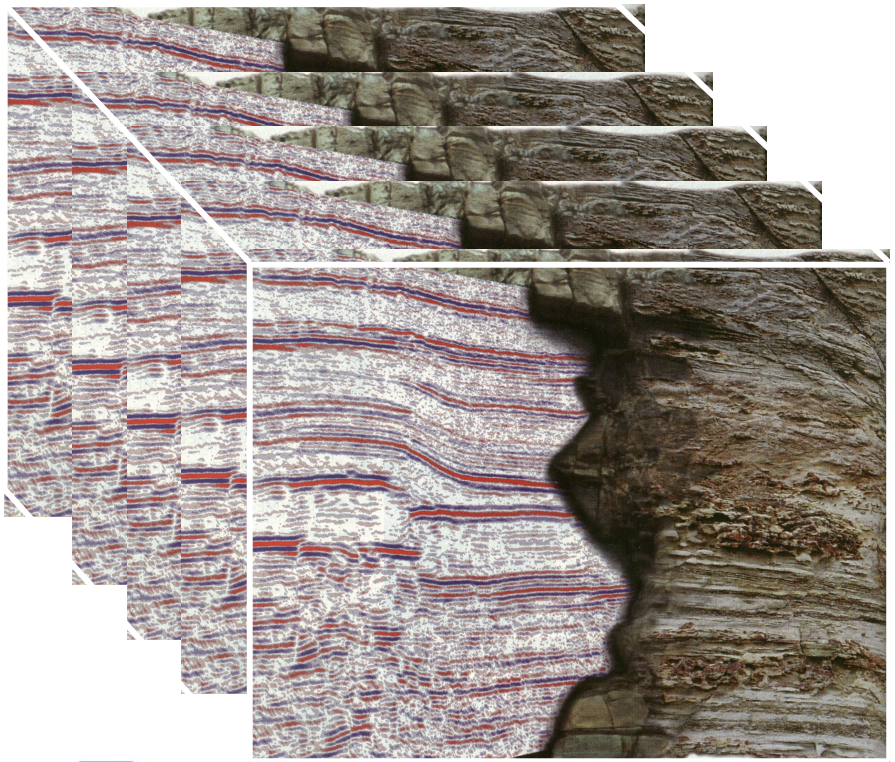
Aquisição Sísmica



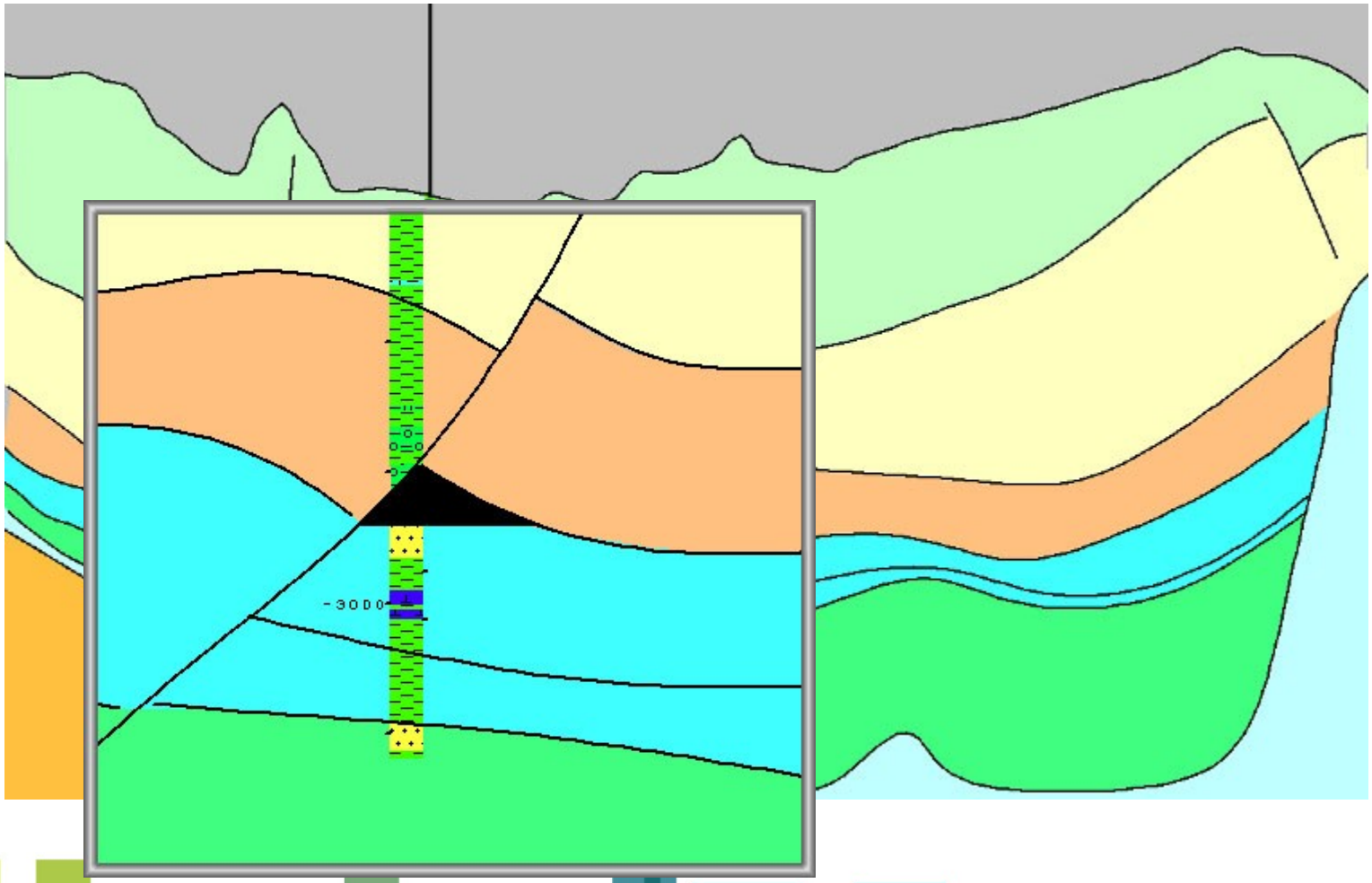
Linhas Sísmicas – Volumes de dados 3D

300 a 500 linhas separadas por 12,5m

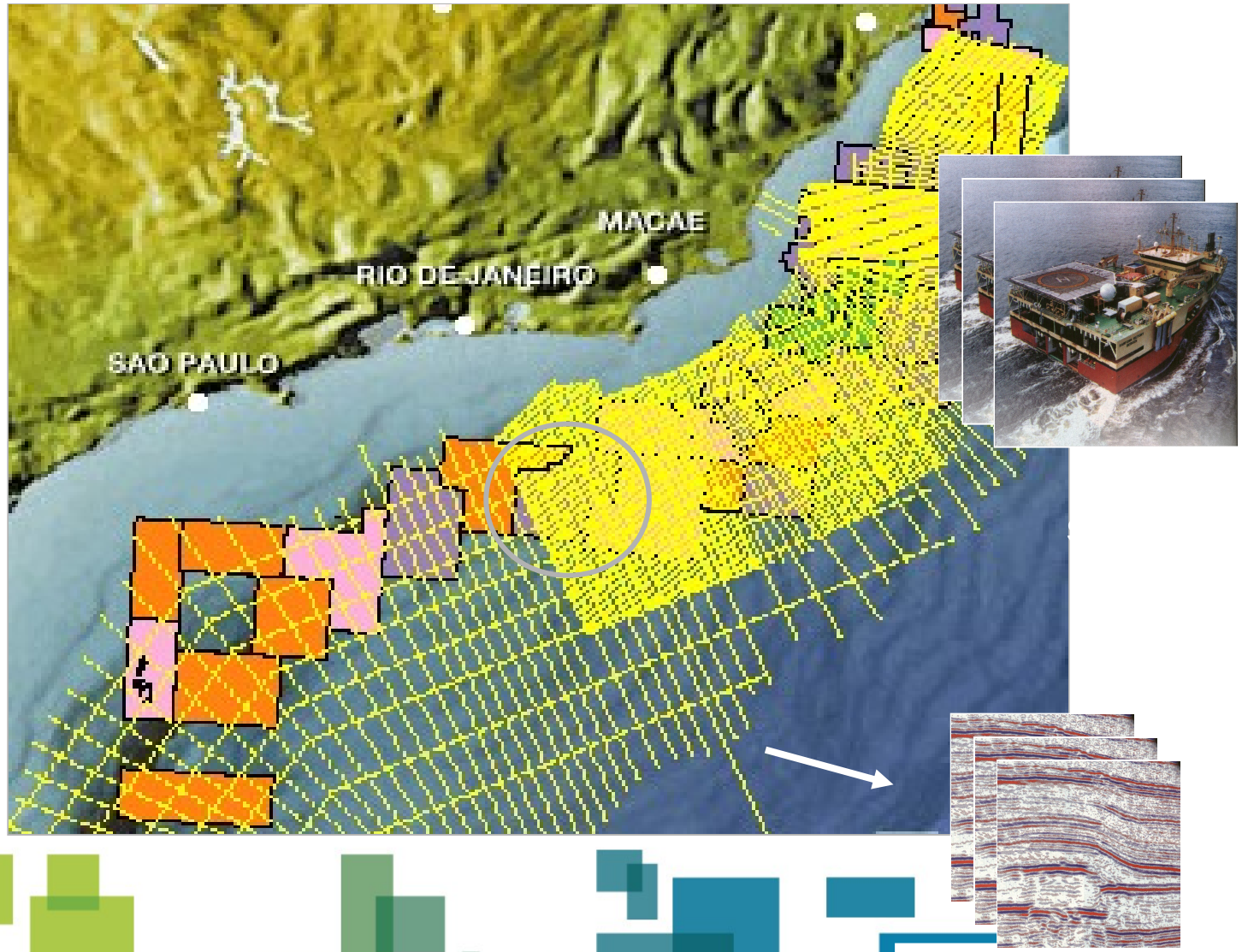




Interpretação Geológica



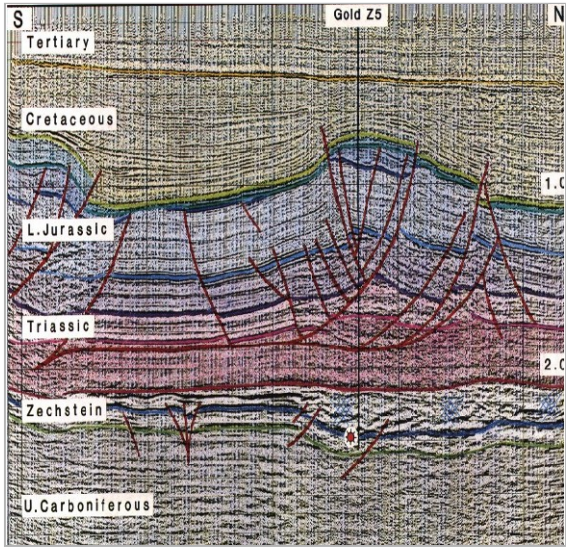
Aquisição de Dados Sísmico 3D

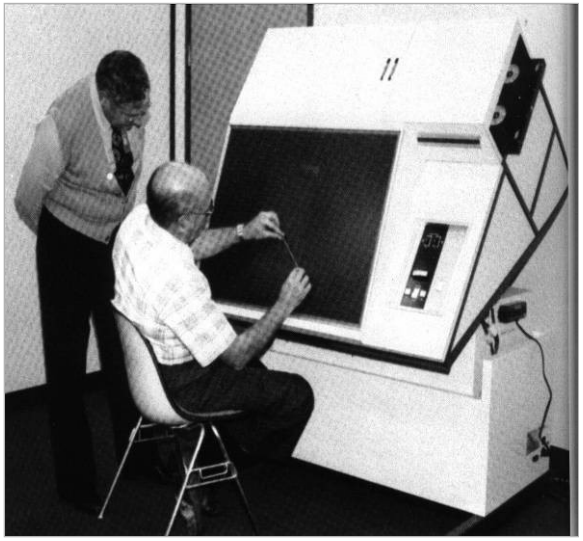


Interpretação Sísmica

Sismogramas, Plotagem, *Displays* CRT

1930 → **1960** → **1980**

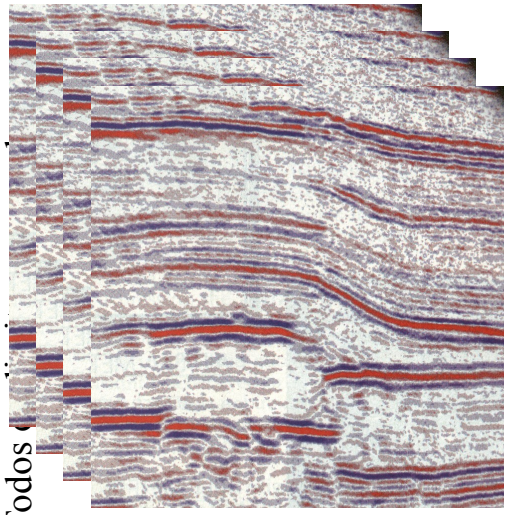




Muito tempo para interpretar uma prospecção

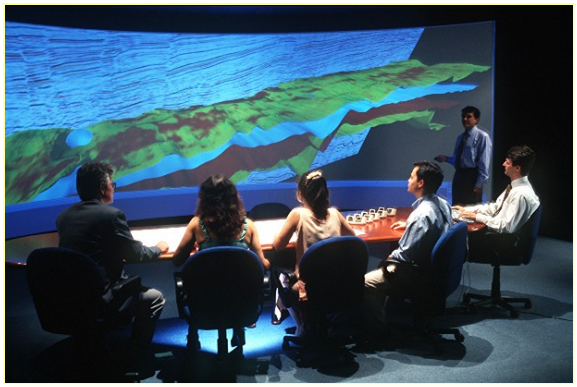
Anos ⇒ **Meses**

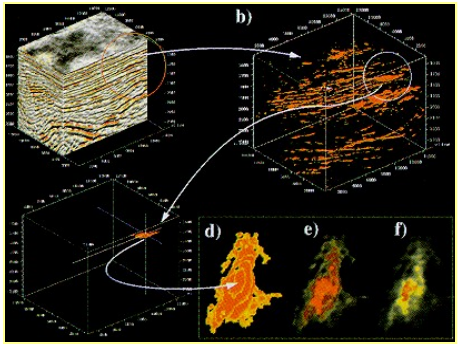
Interpretação Sísmica

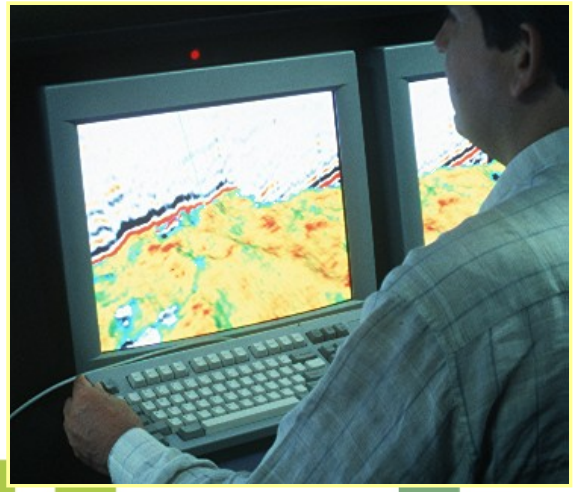


Workstations Gráficas, Interpretação Volumétrica e Centros de Visualização

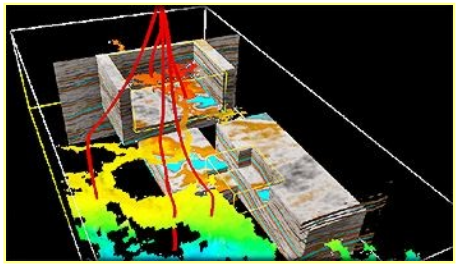
1990 → 1997





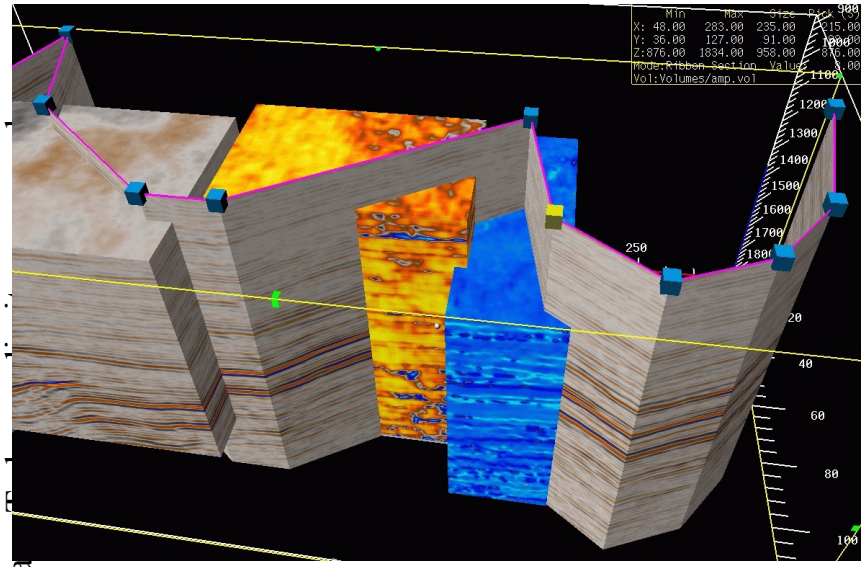


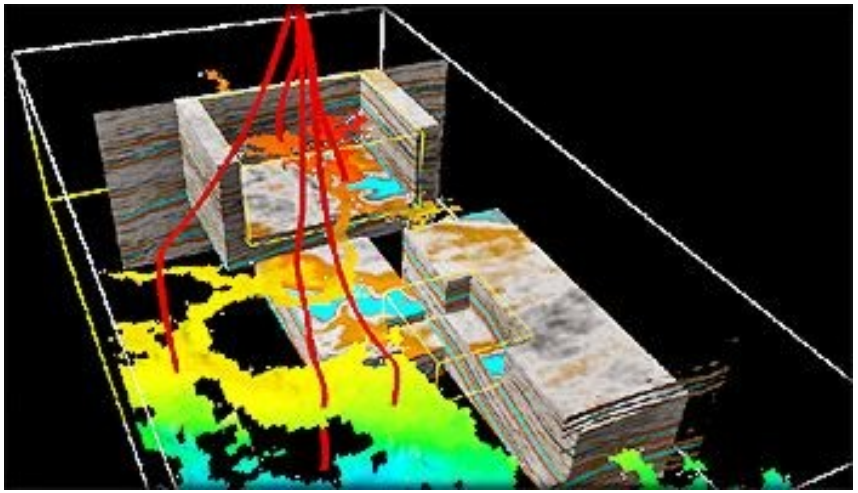
Tempo: semanas

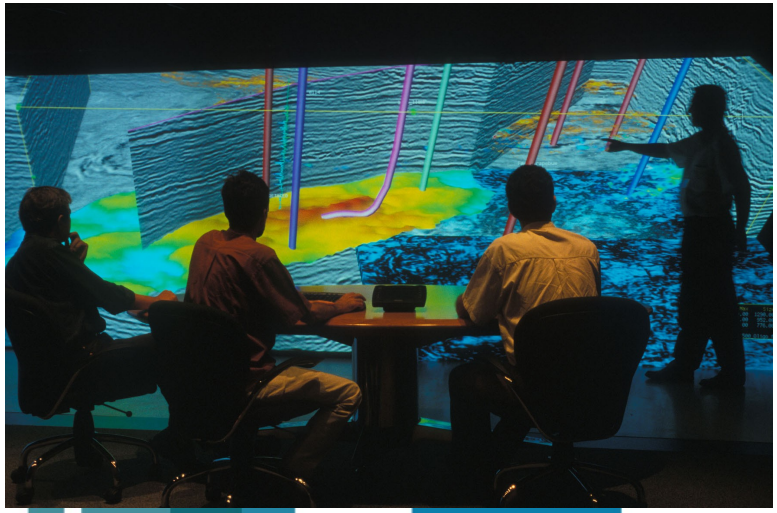


Tempo: meses

Interpretação Sísmica





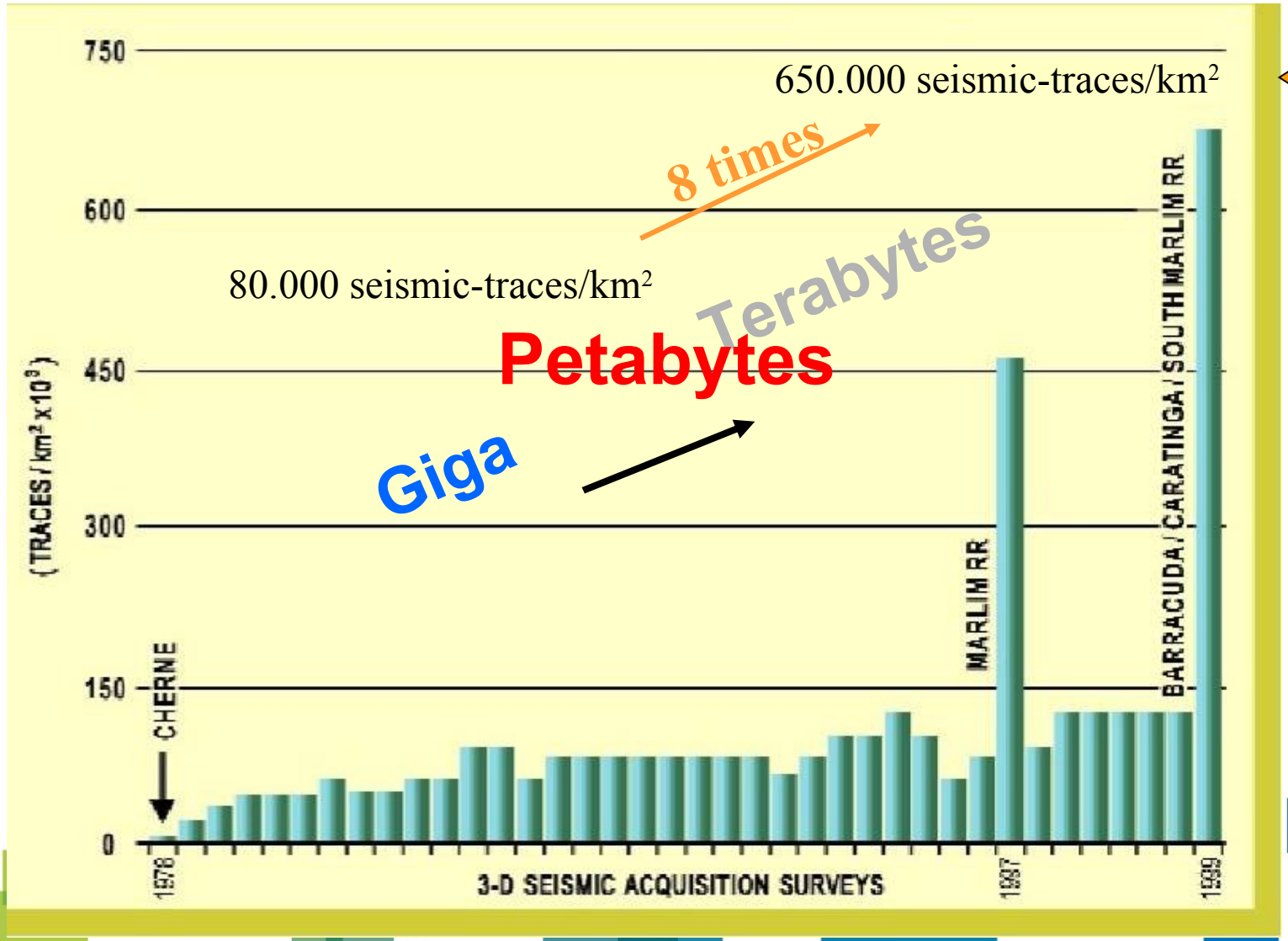


2011

WorkStations Superpoderosas,
Sala de Visualização 3D

Tempo: dias

Crescimento no Volume de Dados 3D



Método Sísmico

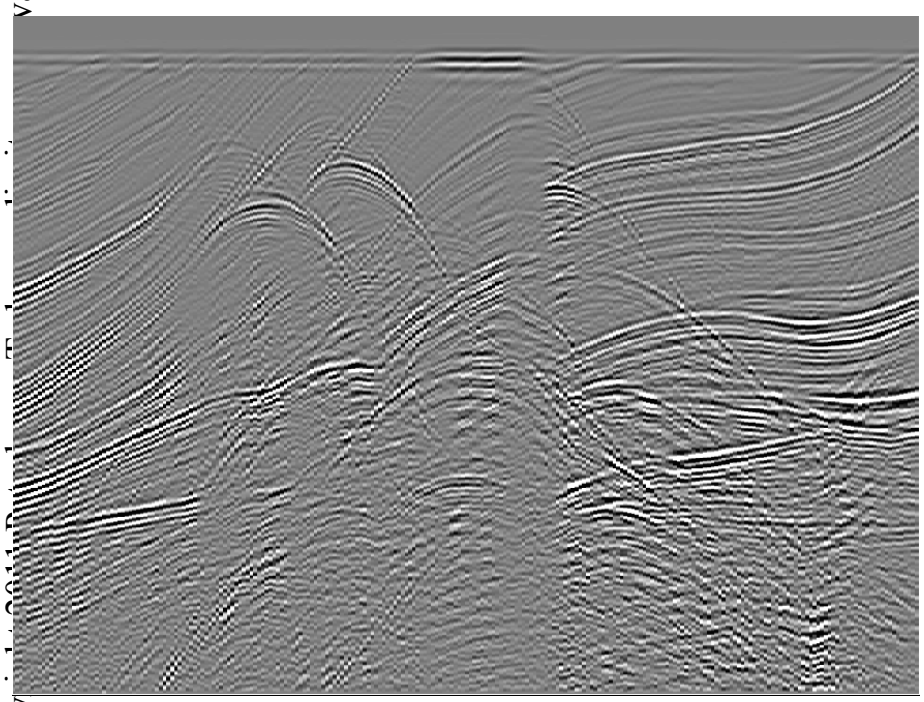
- Serve para?
  - Localizar prováveis reservas de óleo e gás;
  - Ajudar a selecionar os melhores locais para perfurar;
  - Otimizar a produção dos poços.

- Dividido em 3 etapas:
  - Aquisição dos dados;
  - Processamento dos dados;
  - Interpretação dos dados.

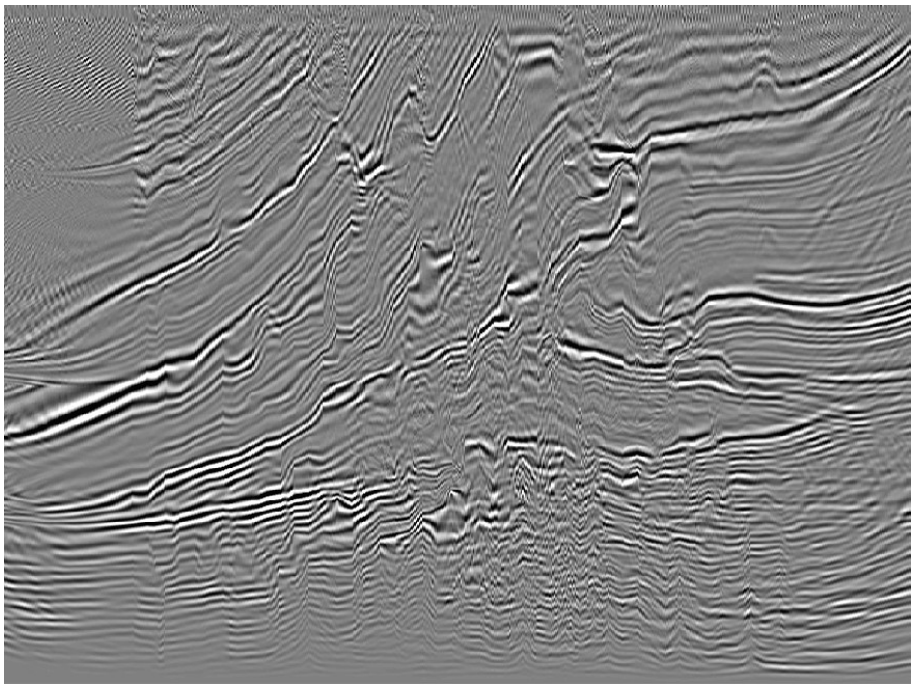
Processamento Sísmico

- Pré-processamento:
  - ordenação;
  - eliminação de ruído;
  - regularização.
- **Análise velocidade e Migração** (+ custosa => PAD);
- Pós-processamento.

Migração Sísmica



Entrada



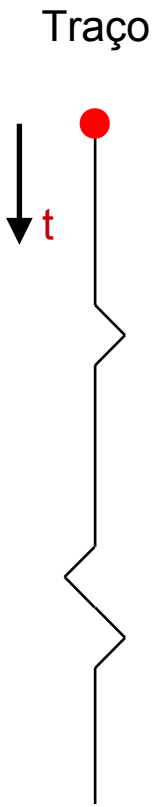
Saída

Tecnologia Geofísica na Petrobras

- Pesquisa de novos métodos e algoritmos geofísicos;
- P&D voltada para as necessidades da empresa;
- Desenvolve e mantém códigos em produção:
  - Clusters executam quase que exclusivamente; software desenvolvido internamente;
  - Migração Kirchhoff em produção desde 1992.

Migração Sísmica

- Para mapear o interior da terra, os registros sísmicos recebidos e gravados em função do tempo devem ser convertidos para registros em função da profundidade;
- As ondas sísmicas possuem velocidades altamente dependentes do meio;
- A profundidade é encontrada a partir do conhecimento das velocidades de propagação e do tempo de trânsito do sinal.



Migração Sísmica

- Processo de produzir uma imagem do subsolo consistente com os dados adquiridos, através do posicionamento das superfícies refletoras;
- É um problema inverso, produz parâmetros do modelo a partir de dados observados;
- Requer múltiplas execuções;
- Grande quantidade de dados (3D).

O Algoritmo

```
Para todos traços de entrada {
    Lê traço de entrada;
    Aplica um filtro multi-frequência;
    Para todas amostras de saída {
        Calcula tempo de trânsito;
        Calcula filtro;
        Interpola bilinearmente;
        Calcula correção de amplitude;
        Acumula a contribuição;
    }
}
```

O Algoritmo

Para *todos traços de entrada {*

Lê traço de entrada;

Aplica um filtro multi-frequência;

Para *todas amostras de saída {*

Calcula tempo de trânsito;

Calcula filtro;

Interpola bilinearmente;

Calcula correção de amplitude;

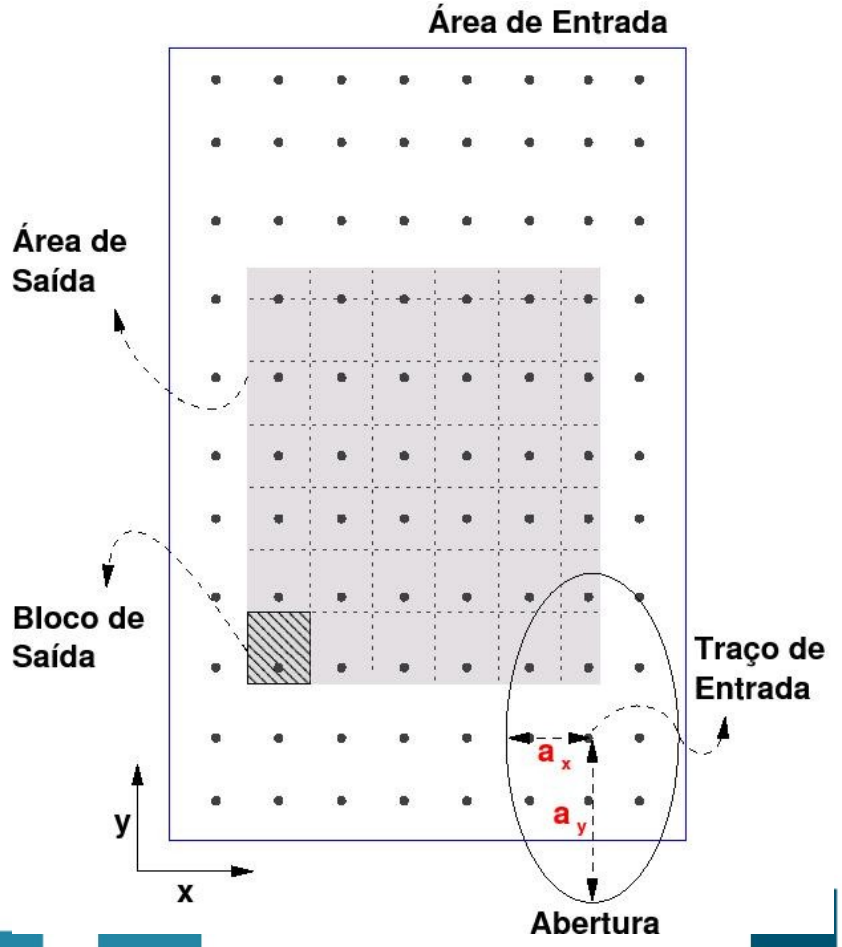
Acumula a contribuição;

}

}

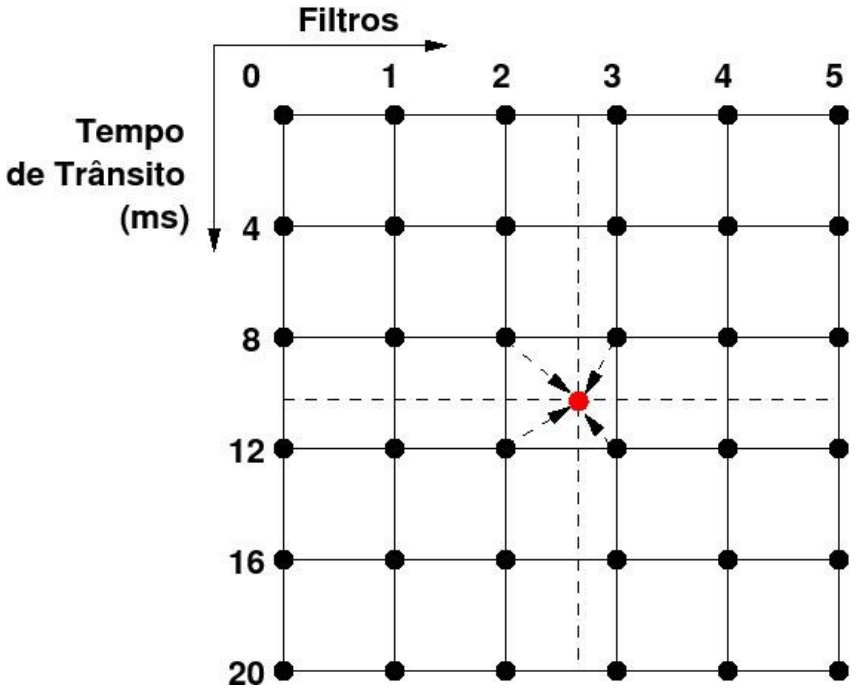
Migração Sísmica

- Paralelismo Mestre/Escravo;
- Blocos de saída são atribuídos dinamicamente.



Interpolação bilinear

- Custo alto;
- Mapear quais amostras de entrada contribuirão para as amostras do traço de saída;
- Amostras em posições discretas e coordenadas em ponto flutuante.



Perfil da Migração de Kirchhoff

- Intensivo em processamento:
  - 75 ops de ponto flutuante por cálculo de amostra.
- Memória:
  - 160 KB por traço processado;
  - 20 MB por bloco de saída;
  - TTT usam o resto da memória (PSDM).
- Paralelismo Mestre/Escravo:
  - Escalabilidade testada até em 8192 CPUs no IBM Blue Gene/L;
  - Balanceamento de carga dinâmico;
  - Tolerância a falhas.
- Integrado a pacotes sísmicos comerciais.

Precisamos de acelerar ainda mais...

- Melhorias no desempenho do hardware podem trazer ganhos importantes, como:
  - Investimentos (dinheiro!!);
  - Custos de data center (importante);
  - Produtividade (muito importante).

- Já possuímos algoritmos melhores para imageamento, mas não temos recursos computacionais suficientes para eles!

Momento de procurar por tecnologias mais rápidas!

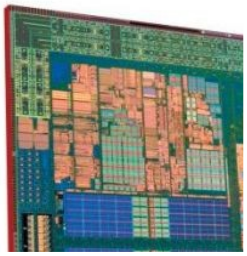
Tecnologias Estudadas

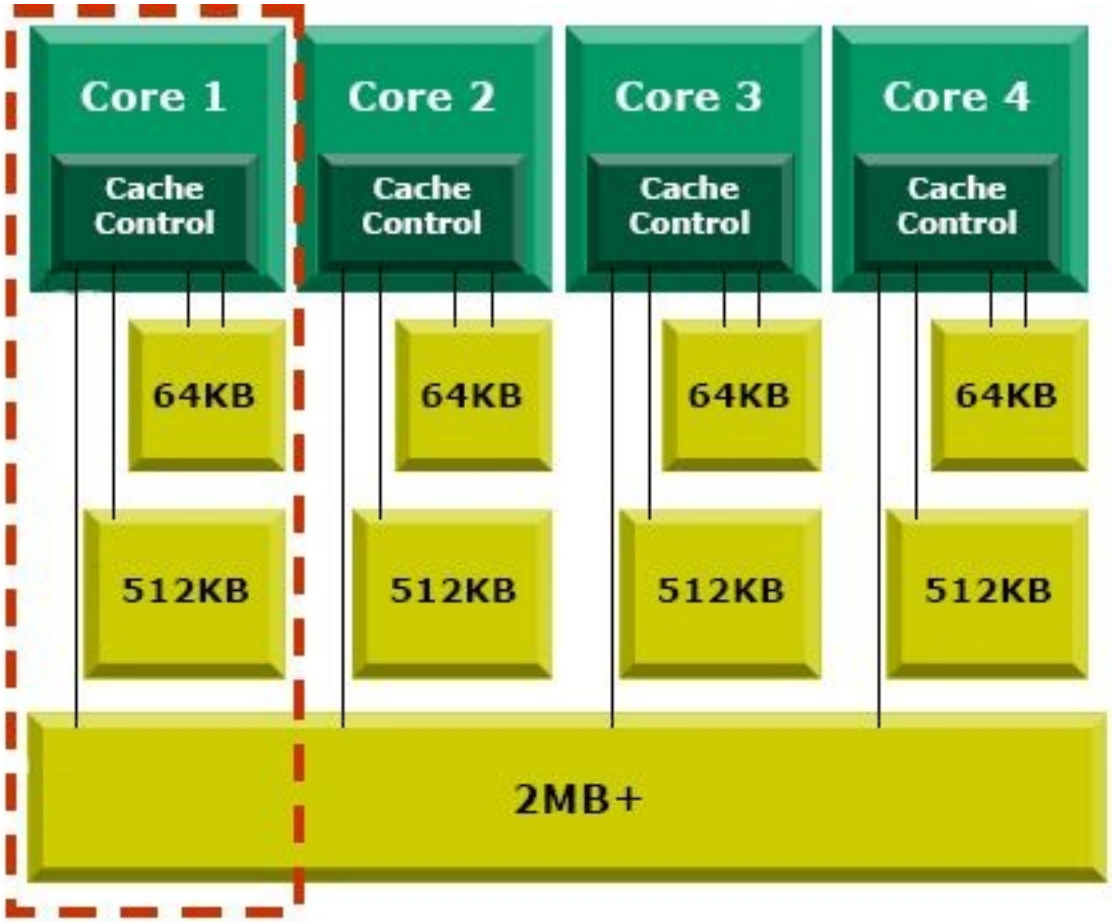
- Multicore tradicional:
  - Intel Xeon, Amd Opteron …
- Cell:
  - PS3 e IBM Blades;
- GPGPU:
  - Nvidia Tesla, GeForce;

Métricas:

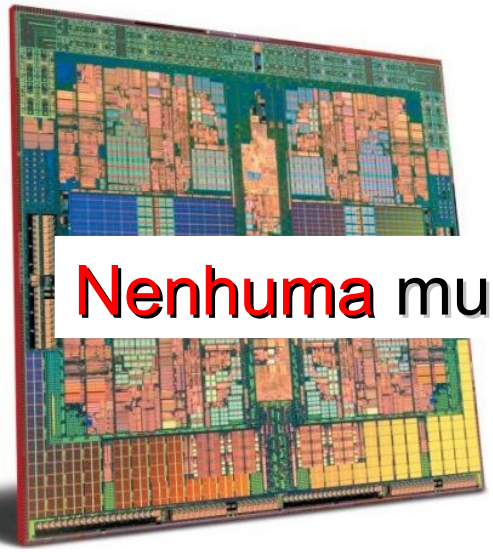
- Milhões de contribuições/seg;
- Milhões de contribuições/Watt;
- Milhões de contribuições/$.

Multicore Tradicional





Multicore Tradicional



Nenhuma mudança no código, um escravo por core!

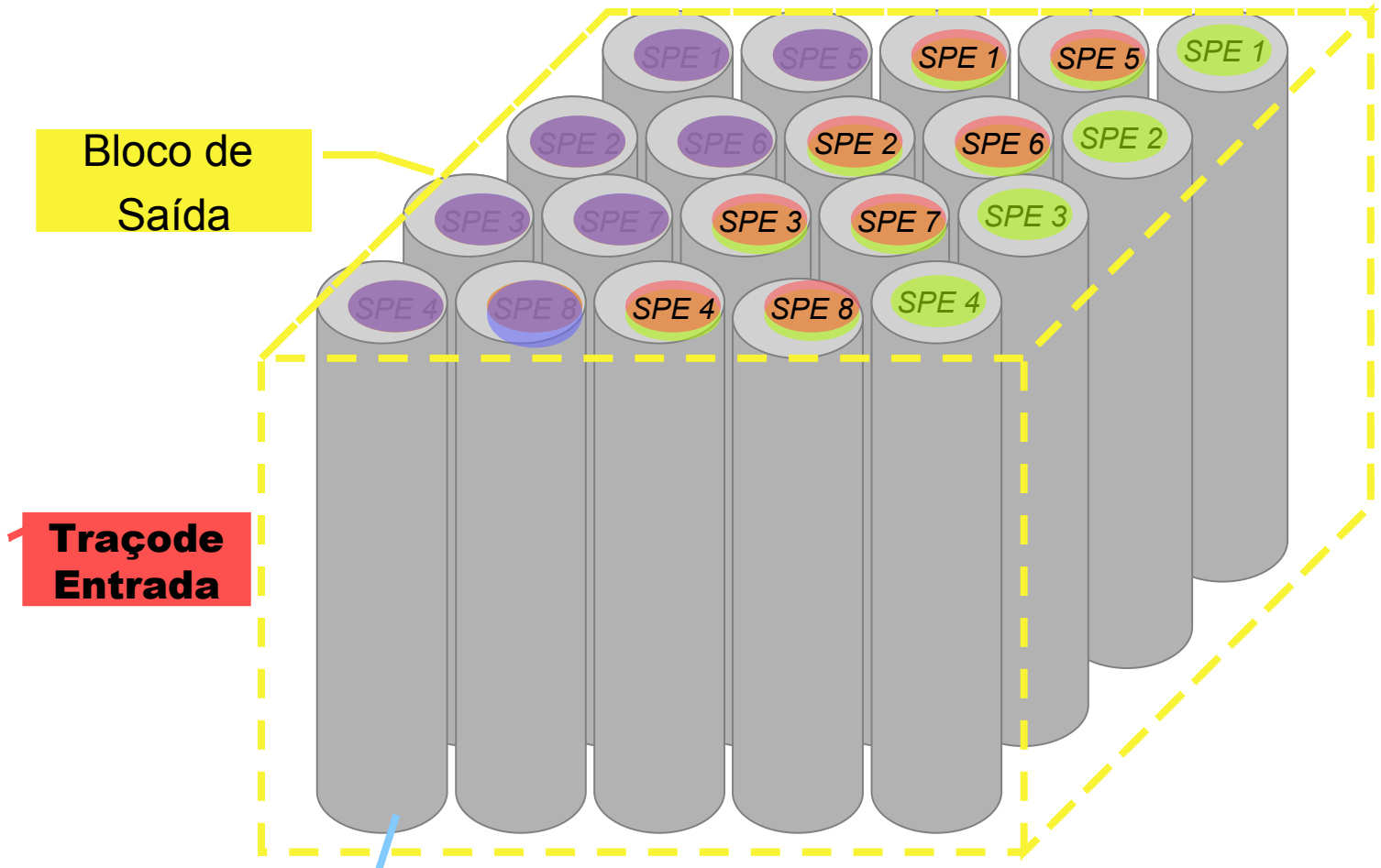
IBM Cell no Sony PlayStation 3

- Heterogêneo → 2 tipos de cores: PPE e SPE;
- 256 KB de memória por SPE (dados e código)!
- Codificação complicada!

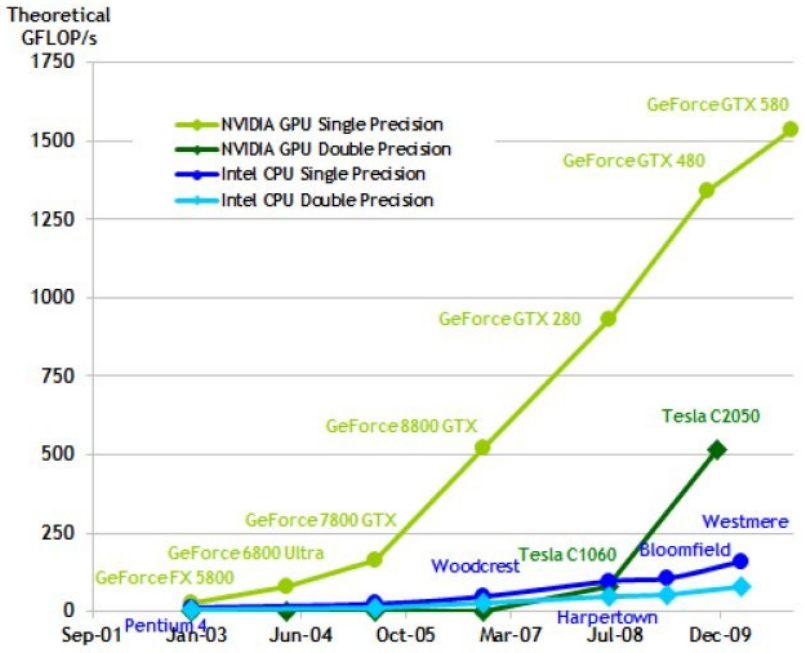
Estratégia de Implementação no Cell

- Um escravo por Cell (PlayStation);
- Decomposição de domínio dos traços de saída entre os SPEs;
- PPE:
  - Responsável pelo bloco de saída;
  - Lê e filtra os traços de entrada.
- Cada SPE:
  - Para cada traço filtrado lido da memória pelo PPE:
    - Lê um dos traços de saída da memória PPE, enquanto;
    - Calcula a contribuição do traço de saída corrente, enquanto;
    - Armazena o traço de saída anterior na memória do PPE.

Paralelismo Intra Cell



GPGPU



- 1+ Tflop de pico;
- 3 GB / 6 GB por GPU;
- Interpolação em HW!

- Programação diferente (mais difícil?);
- Transferência de dados explícita (+ custosa);
- Hierarquia de memória mais complicada;
- Divergências? NÃO!

Como Programar GPUs?

- Modelo SIMT → MUITAS threads!
- Modelo de programação diferente;
- Heterogeneidade (e as CPUs?) ;
- Linguagens:
  - CUDA:
    - C com extensões;
    - Bem documentado;
    - Suporte da Nvidia.
  - OpenCL:
    - Padrão aberto;
    - Serve para CPUs, GPUs, FPGAs...
    - Avançando lentamente.

Estratégia de Implementação na GPU

- CPU:
  - Lê e filtra traços de entrada;
  - Copia o traço filtrado para a GPU;
  - Dispara o kernel na GPU.
- GPU:
  - Mantém o bloco de saída na memória da GPU;
  - Calcula as contribuições do traço de entrada corrente para todas as amostras do bloco de saída:
    - Gera 1 milhão de threads independentes e utiliza interpolação em hardware (textura).
  - Múltiplos escravos por GPU;
  - Filtragem de traços filtrados é feita assincronamente.

Sobrecarga e Sobreposição!

Uma CPU vs Uma GPU

Componente	Núcleo CPU	CPU+GPU Síncrono	CPU+GPU Cooperação
Filtragem	636	636	659
Contribuições (Ganho)	41.324	466 (89)	
Resto	48	48	48
Total (Ganho)	42.008	1.150 (36)	707 (59)

Tempo de execução (seg), um bloco de saída

Múltiplas CPUs vs Uma GPU

Núcleos CPU (blocos de saída)	1	2	3	4
Sem GPU	48	96	145	193
Uma GPU **(Ganho)**	2.873 **(59)**	3.865 **(39)**	3.914 **(27)**	3.923 **(20)**

Milhões de amostras contribuídas / segundo (Msamples/s)

Ganhos no Cluster Completo

- Cluster com 32 nós com 8 núcleos e 2 GPUs.

Sem GPUs \ Com GPUs	64	128	192
64	59		
128	29	39	
192	20	26	27
256	15	20	20

Não podemos desprezar os núcleos CPU ociosos!

Múltiplas Configurações e Utilização Eficiente dos Recursos (1)

- Como utilizar todos os recursos disponíveis de maneira eficiente?
- Como utilizar eficientemente múltiplas configurações heterogêneas?
- Como fazer isso de maneira sem ter que reprogramar a cada nova configuração adquirida?

Múltiplas Configurações e Utilização Eficiente dos Recursos (2)

- Problema: capacidades de processamento diferentes;
- O recurso mais lento atrasa toda a execução quando recebe um bloco de saída;
- Solução: colocar todos os recursos computacionais para gerar contribuição de um mesmo bloco de saída. Como?
- Vários traços de entrada são processados simultaneamente.

Múltiplas Configurações e Utilização Eficiente dos Recursos (3)

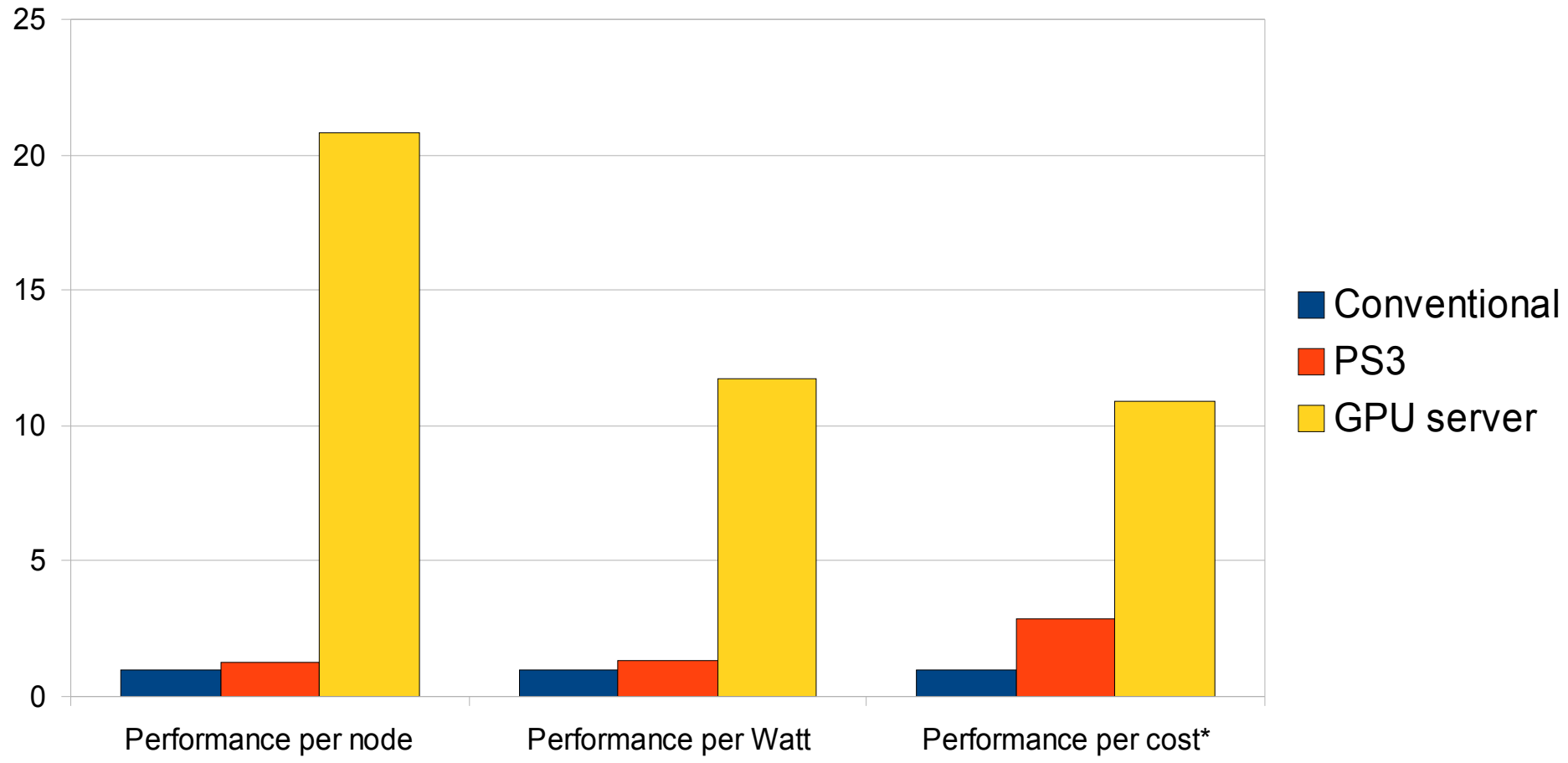
- Problema: existe uma condição de corrida → cada amostra de saída somente pode receber a contribuição de um traço de entrada por vez;
- Solução: vários traços de entrada são migrados simultaneamente em cópias diferentes do bloco de saída;
- Ao final, os resultados de cada cópia são agregados para gerar o resultado final.

Múltiplas Configurações e Utilização Eficiente dos Recursos (4)

- Foi utilizado um escalonamento dinâmico de grão fino, por traços de entrada, e com isso:
  - várias unidades de processamento podem ser utilizadas para um mesmo bloco;
  - processam traços na velocidade que cada uma pode alcançar;
  - não deixam recursos ociosos.

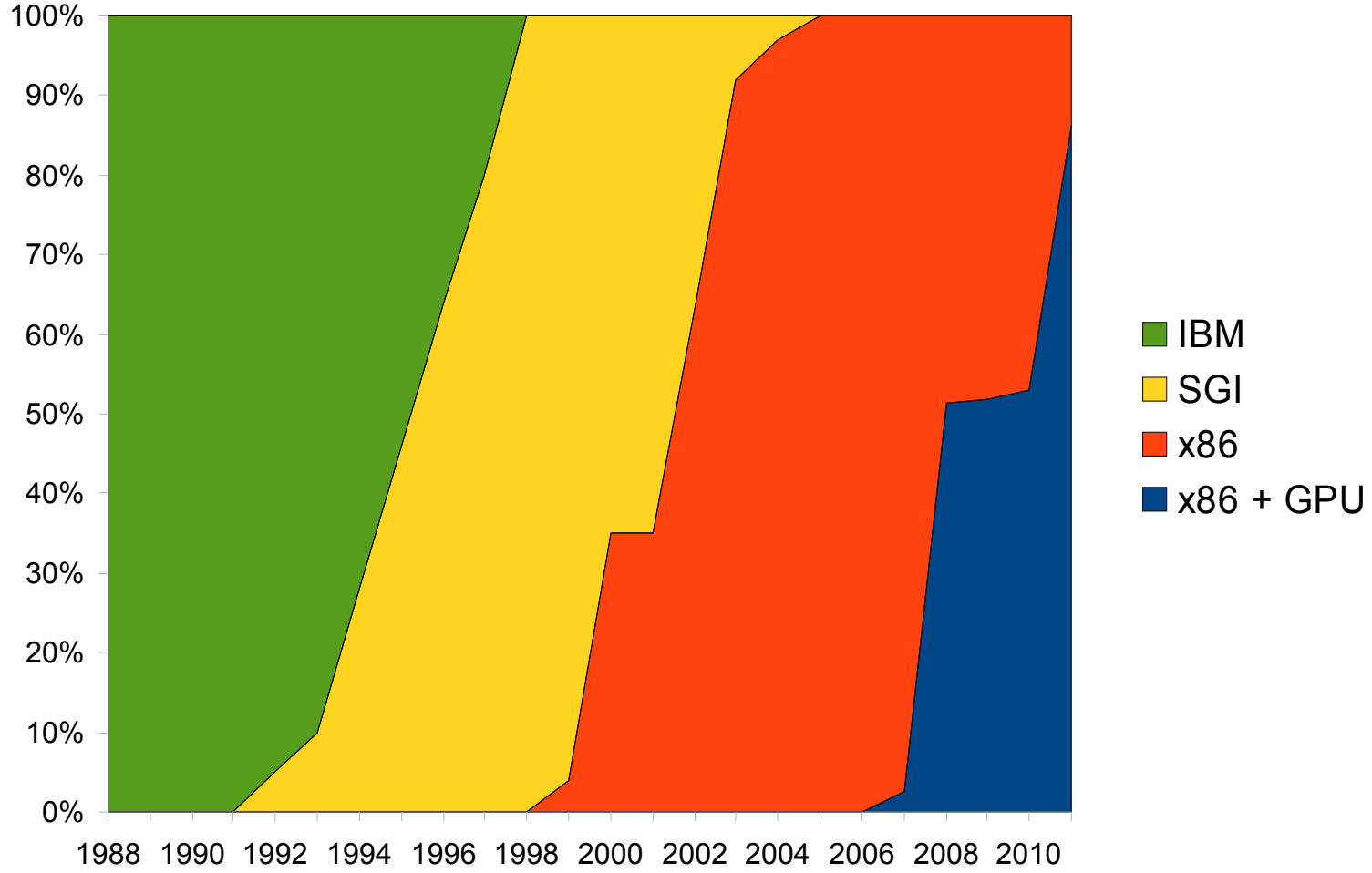
- O *overhead* de agregar os resultados é proporcionalmente muito pequeno (apenas uma soma) e pode ser otimizado (vetorização e escrita alinhada).

Resultados da Migração



*custo = Preço de aquisição + energia e resfriamento por 4 anos

Evolução na Capacidade de Processamento



Produção - 2006

- 5.000 núcleos de CPU;
- 1.000 execuções da migração Kirchhoff.

Execuções mais exigentes	Núcleos/ exec (média)	Dia/exec (média)
10	787	22
100	656	10
1000	185	2

Produção - 2008

- 14.000 núcleos de CPU;
- 3.000 execuções da migração Kirchhoff.

Execuções mais exigentes	Núcleos/ exec (média)	Dia/exec (média)
10	950	36
100	815	17
1000	196	3

Produção - 2011

- 20.000 núcleos de CPU;
- 1.462 GPUs;
- 4.000 execuções da migração Kirchhoff.

Execuções mais exigentes	Núcleos /exec (média)	Dia/exec (média)
10	2116	20
100	1343	6,5
1000	717	1,2

Imageamento usando GPUs

- Cinco clusters de GPU:
  - 3 na produção e 2 para pesquisa.
- Algoritmos:
  - Regularização de dados:
    - RMIL.
  - KTM and Análise de Velocidade:
    - TTI e VTI (anisiotropia).
  - Migração Kirchhoff em Profundidade;
  - One Way Wave Equation Migration;
  - Migração Reversa no Tempo.

Trabalhos em Andamento

- OpenCL:
  - Open Computing Language;
  - padrão aberto para CPUs, GPUs e aceleradores.
- GPUs AMD:
  - 2.7 Tflops de desempenho.
- Intel® Many Integrating Core Architecture.

Trabalhos Futuros

- Processadores ARM:
  - Estão em todo lugar (tables, smarphones);
  - Baixo consumo de energia e preço!
- Processadores Integrados a GPUs:
  - AMD Fusion (x86 + AMD GPU);
  - Nvidia Tegra / Projeto Denver (ARM + NVIDIA GPU).
- Características:
  - Mesma memória CPU e GPU;
  - Código mais simples, menos cópias;
  - Quantidade de memória independente.

Tablet based on Nvidia Tegra 2: www.dell.com

Conclusões

- Utilização em produção de clusters heterogêneos;
- Redução de tempo em 20x sobre um cluster sem GPUs;
- Grifo01 com 188 GPUs = 15600 núcleos Xeon* em 10x menos espaço;
- Utilização eficiente de recursos heterogêneos

  (Sobrecarga e Sobreposição);

*Xeon E5410

Conclusões

- Resultados excepcionais para a utilização de aceleradores em processamento sísmico;
- Não é mais possível fugir desse tipo de ambiente;
- CPUs cada vez mais integradas com os aceleradores;
- Quem sabe teremos alternativas para o x86...

Agradecimentos

- Petrobras pela oportunidade de realização e divulgação das pesquisas realizadas;
- AMD, Intel, Nvidia e IBM pelo suporte contínuo e disponibilidade antecipada de protótipos;
- NCSA: *National Center for Supercomputing Applications* pelo acesso antecipado a um cluster de GPU.

Obrigado!

www.petrobras.com.br

Dúvidas?

Contato: thiagoxt@petrobras.com.br