Compiladores Comensais

Pedro Bruel, Prof. Dr. Alfredo Golman - {phrb, gold}@ime.usp.br

IME-USP

Laboratório de Sistemas de Software

Índice

Contexto

Bosboom, Jeffrey, et al. *"StreamJIT: a commensal compiler for high-performance stream programming."* **Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications**.

- Compilação Comensal;
- Compilador Comensal para a JVM.

Contexto

Bosboom, Jeffrey, et al. *"StreamJIT: a commensal compiler for high-performance stream programming."* **Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications**.

- Compilação Comensal;
- Compilador Comensal para a JVM.

Rompf, Tiark, et al. *"Go Meta! A Case for Generative Programming and DSLs in Performance Critical Systems."* **1st Summit on Advances in Programming Languages (2015): 238.**

- Discussão sobre DSLs, bibliotecas e meta-programação;
- Arcabouço Delite.

Contexto

Linguagens de Domínio Específico vs. Bibliotecas

Linguagens de Domínio Específico

- (+) Otimizações específicas;
- (+) Abstrações e interfaces do domínio.

- (-) Familiaridade;
- (-) **Custo de implementação**:
  (compiladores, depuradores...)

Contexto

Linguagens de Domínio Específico vs. Bibliotecas

Linguagens de Domínio Específico

- (+) Otimizações específicas;
- (+) Abstrações e interfaces do domínio.

- (-) Familiaridade;
- (-) **Custo de implementação**:
  (compiladores, depuradores...)

- Halide (Processamento de Imagens):
  halide-lang.org
- SuperCollider (Processamento de Áudio):
  supercollider.github.io
- StreamIt (Streaming Systems):
  groups.csail.mit.edu/cag/streamit

Contexto

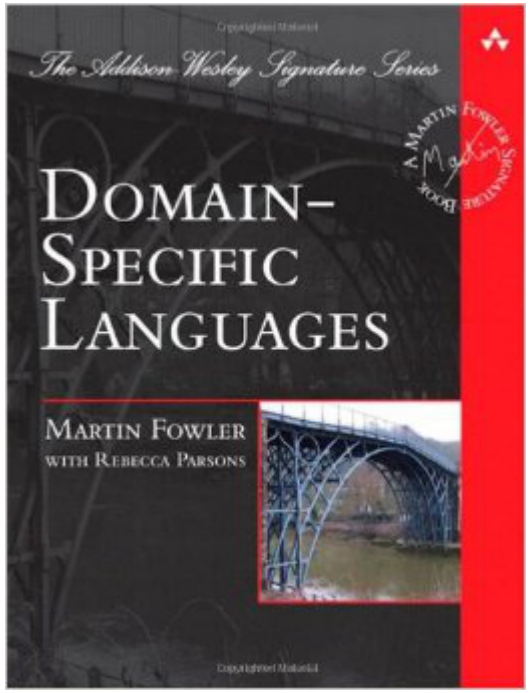
Linguagens de Domínio Específico vs. Bibliotecas

Linguagens de Domínio Específico

- (+) Otimizações específicas;
- (+) Abstrações e interfaces do domínio.

- (-) Familiaridade;
- (-) **Custo de implementação**:
  (compiladores, depuradores...)

- Halide (Processamento de Imagens):
  halide-lang.org
- SuperCollider (Processamento de Áudio):
  supercollider.github.io
- StreamIt (Streaming Systems):
  groups.csail.mit.edu/cag/streamit



Contexto

Linguagens de Domínio Específico vs. Bibliotecas

Bibliotecas

- (+) **Custo de implementação**;
- (+) Ferramentas de linguagem.

- (-) Otimizações específicas ao domínio;
- (-) Abstrações e interfaces.

Contexto

Linguagens de Domínio Específico vs. Bibliotecas

Bibliotecas

- (+) **Custo de implementação**;
- (+) Ferramentas de linguagem.

- (-) Otimizações específicas ao domínio;
- (-) Abstrações e interfaces.

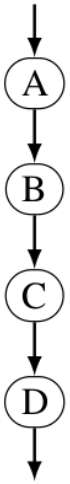
- ImageMagick (Processamento de Imagens): **imagemagick.org**
- Aquila (Processamento de Áudio): **aquila-dsp.org**
- LAPACK (Álgebra Linear): **netlib.org/lapack/**
- ...

StreamJIT

- Implementação em Java da linguagem StreamIt. (**groups.csail.mit.edu/cag/streamit**)
- **github.com/jbosboom/streamjit**

StreamJIT

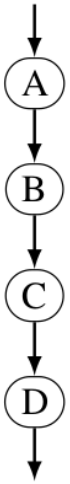
- Implementação em Java da linguagem StreamIt. (**groups.csail.mit.edu/cag/streamit**)
- **github.com/jbosboom/streamjit**



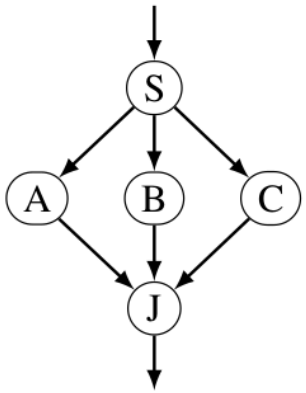
(a) Pipelines compose one-to-one elements (filters, pipelines or splitjoins) by connecting each element's output to the input of the next element.

StreamJIT

- Implementação em Java da linguagem StreamIt. (**groups.csail.mit.edu/cag/streamit**)
- **github.com/jbosboom/streamjit**



(a) Pipelines compose one-to-one elements (filters, pipelines or splitjoins) by connecting each element's output to the input of the next element.



(b) Splitjoins compose a splitter, a joiner, and one or more one-to-one elements (filters, pipelines or splitjoins) by connecting the splitter's outputs to the inputs of the branches and the outputs of the branches to the joiner's input.

Compiladores Comensais

Proposta

A técnica de Compilação Comensal propõe:

Compiladores Comensais

Proposta

A técnica de Compilação Comensal propõe:

- Diminuir o esforço de implementação;

Compiladores Comensais

Proposta

A técnica de Compilação Comensal propõe:

- Diminuir o esforço de implementação;
- Implementar otimizações específicas ao domínio;

Compiladores Comensais

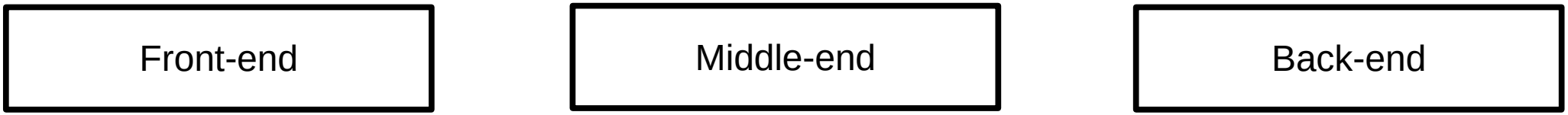
Proposta

A técnica de Compilação Comensal propõe:

- Diminuir o esforço de implementação;
- Implementar otimizações específicas ao domínio;
- Aproveitar as ferramentas de uma linguagem já estabelecida.

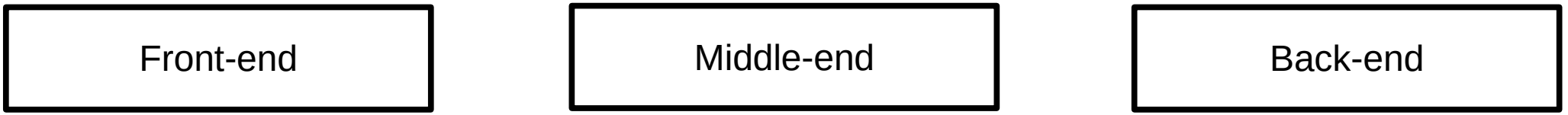
Compiladores Comensais

Visão Geral



Compiladores Comensais

Visão Geral



- Código implementado na linguagem hospedeira;
- Analisador Sintático e outras ferramentas;
- Compilado como uma biblioteca.

Compiladores Comensais

Visão Geral

Front-end

Middle-end

Back-end

- Código implementado na linguagem hospedeira;
- Analisador Sintático e outras ferramentas;
- Compilado como uma biblioteca.

- Otimizações “genéricas” são feitas pelo compilador da linguagem hospedeira;
- Otimizações específicas do domínio definem um espaço de busca;
- Portanto, podem ser feitas por um **auto-tuner**.

Compiladores Comensais

Visão Geral

Front-end

- Código implementado na linguagem hospedeira;
- Analisador Sintático e outras ferramentas;
- Compilado como uma biblioteca.

Middle-end

- Otimizações "genéricas" são feitas pelo compilador da linguagem hospedeira;
- Otimizações específicas do domínio definem um espaço de busca;
- Portanto, podem ser feitas por um **auto-tuner**.

Back-end

- Mecanismos de geração de código;
- Programa implementado na linguagem hospedeira;
- *Profiler, debugger.*


```
public abstract class Filter<I, O>
extends Worker<I, O>
implements OneToOneElement<I, O> {
    public Filter(int popRate, int pushRate);
    public Filter(int popRate, int pushRate,
                  int peekRate);
    public Filter(Rate popRate, Rate pushRate,
                  Rate peekRate);

    public abstract void work();

    protected final I peek(int position);
    protected final I pop();
    protected final void push(O item);
}
```


```
public abstract class Filter<I, O>
extends Worker<I, O>
implements OneToOneElement<I, O> {
    public Filter(int popRate, int pushRate);
    public Filter(int popRate, int pushRate,
                  int peekRate);
    public Filter(Rate popRate, Rate pushRate,
                  Rate peekRate);

    public abstract void work();

    protected final I peek(int position);
    protected final I pop();
    protected final void push(O item);
}
```

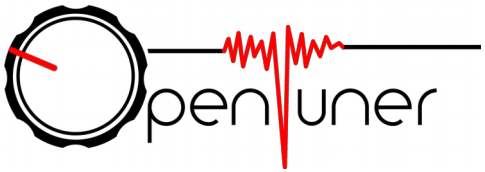
- Implementações das abstrações em Java;
- *Pop, push, peek;*
- Uso de Tipos Genéricos da JVM.

Middle-end

- Otimizações “genéricas” são feitas pela JVM;
- Otimizações de domínio são feitas no nível da RI;

Middle-end

- Otimizações “genéricas” são feitas pela JVM;
- Otimizações de domínio são feitas no nível da RI;
- Otimizações específicas são feitas pelo **OpenTuner**.



Back-end

- `MethodHandle`;
- Geração de *bytecode*;

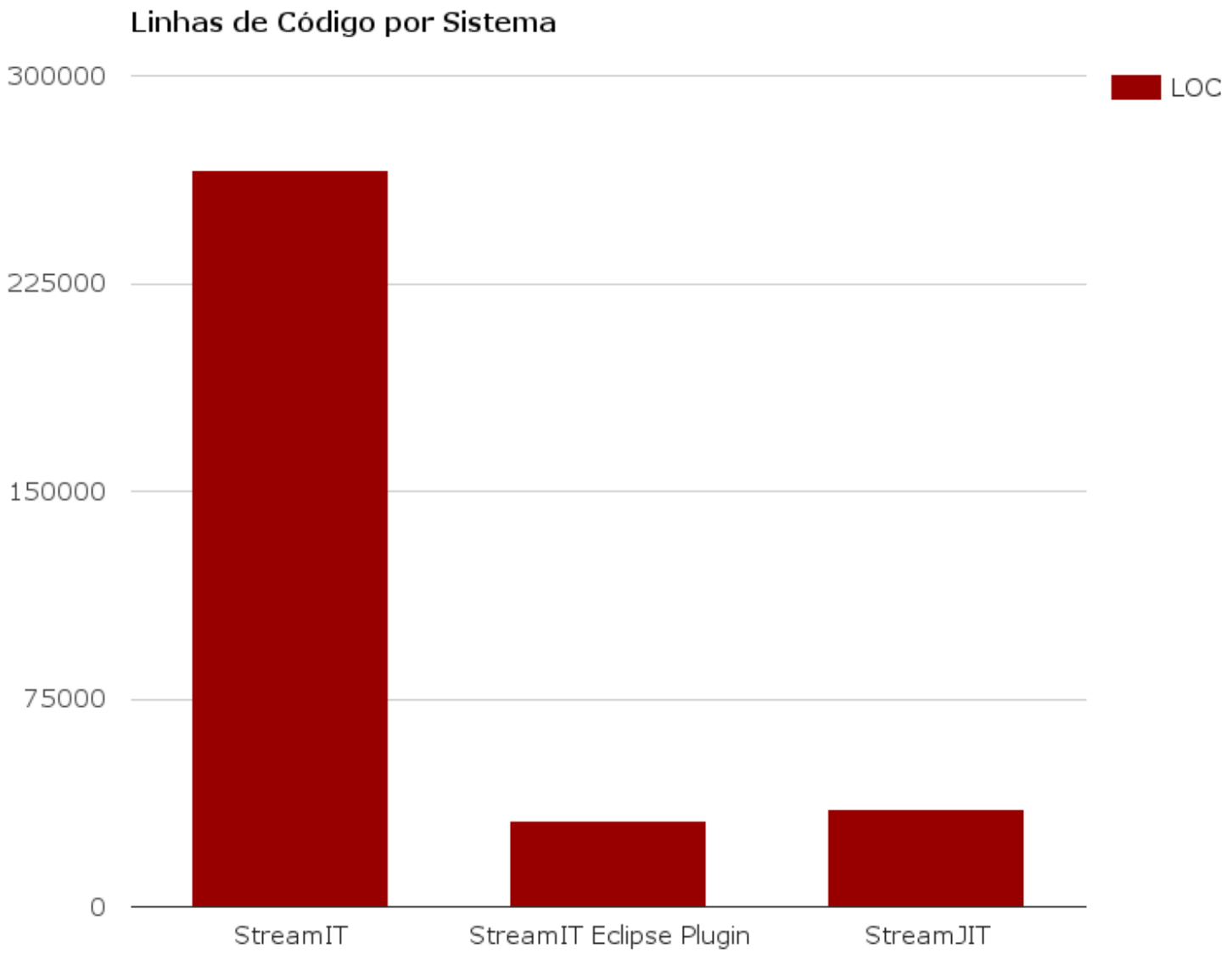
Back-end

- `MethodHandle`;
- Geração de *bytecode*;

```
private static void loop(MethodHandle loopBody,
        int begin, int end, int increment) throws
        Throwable {
    for (int i = begin; i < end; i += increment)
    loopBody.invokeExact(i);
}
```

Compiladores Comensais

Esforço de Implementação



benchmark	StreamJIT	StreamIt	relative perf
Beamformer	2,320,186	1,204,215	1.9
BitonicSort	9,771,987	6,451,613	1.5
ChannelVocoder	551,065	796,548	0.7
DCT	23,622,047	6,434,316	3.7
DES	17,441,860	6,469,003	2.7
FFT	25,210,084	2,459,016	10.3
Filterbank	924,499	1,785,714	0.5
FMRadio	2,272,727	2,085,143	1.1
MPEG2	32,258,065	-	-
Serpent	2,548,853	6,332,454	0.4
TDE-PP	12,605,042	2,357,564	5.3
Vocoder	406,394	-	-

Figure 16: Single-node 24-core throughput comparison, in outputs per second. Relative performance is StreamJIT throughput divided by StreamIt throughput. StreamIt fails to compile MPEG2 and Vocoder.

Após 12h de *tuning*!

benchmark	StreamJIT	StreamIt	relative perf
Beamformer	2,320,186	1,204,215	1.9
BitonicSort	9,771,987	6,451,613	1.5
ChannelVocoder	551,065	796,548	0.7
DCT	23,622,047	6,434,316	3.7
DES	17,441,860	6,469,003	2.7
FFT	25,210,084	2,459,016	10.3
Filterbank	924,499	1,785,714	0.5
FMRadio	2,272,727	2,085,143	1.1
MPEG2	32,258,065	-	-
Serpent	2,548,853	6,332,454	0.4
TDE-PP	12,605,042	2,357,564	5.3
Vocoder	406,394	-	-

Figure 16: Single-node 24-core throughput comparison, in outputs per second. Relative performance is StreamJIT throughput divided by StreamIt throughput. StreamIt fails to compile MPEG2 and Vocoder.

Após 12h de *tuning*!

benchmark	StreamJIT	StreamIt	relative perf
Beamformer	2,320,186	1,204,215	1.9
BitonicSort	9,771,987	6,451,613	1.5
ChannelVocoder	551,065	796,548	0.7
DCT	23,622,047	6,434,316	3.7
DES	17,441,860	6,469,003	2.7
FFT	25,210,084	2,459,016	10.3
Filterbank	924,499	1,785,714	0.5
FMRadio	2,272,727	2,085,143	1.1
MPEG2	32,258,065	-	-
Serpent	2,548,853	6,332,454	0.4
TDE-PP	12,605,042	2,357,564	5.3
Vocoder	406,394	-	-

Em média, 2.8x de *speedup*.

Figure 16: Single-node 24-core throughput comparison, in outputs per second. Relative performance is StreamJIT throughput divided by StreamIt throughput. StreamIt fails to compile MPEG2 and Vocoder.

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;
- Implementa otimizações de baixo nível específicas ao domínio;

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;
- Implementa otimizações de baixo nível específicas ao domínio;
- Reaproveita ferramentas de *profiling* e depuração;

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;
- Implementa otimizações de baixo nível específicas ao domínio;
- Reaproveita ferramentas de *profiling* e depuração;
- Reaproveita as abstrações de linguagens estabelecidas;

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;
- Implementa otimizações de baixo nível específicas ao domínio;
- Reaproveita ferramentas de *profiling* e depuração;
- Reaproveita as abstrações de linguagens estabelecidas;
- **Compilação pode demorar, devido ao uso de auto-tuners.**

Conclusão

Técnica de Compilação Comensal

- Diminui esforço de implementação;
- Implementa otimizações de baixo nível específicas ao domínio;
- Reaproveita ferramentas de *profiling* e depuração;
- Reaproveita as abstrações de linguagens estabelecidas;
- **Compilação pode demorar, devido ao uso de auto-tuners.**
  - Auto-tuning Online;
  - Auto-tuning Distribuído;

Referências

Bosboom, Jeffrey, et al. *"StreamJIT: a commensal compiler for high-performance stream programming."* **Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications**.

- **github.com/jbosboom/streamjit**

Rompf, Tiark, et al. *"Go Meta! A Case for Generative Programming and DSLs in Performance Critical Systems."* **1st Summit on Advances in Programming Languages (2015): 238.**

Compiladores Comensais

Pedro Bruel, Prof. Dr. Alfredo Golman - {phrb, gold}@ime.usp.br

IME-USP

Laboratório de Sistemas de Software