citation count: 32

Language Support for Regions

David Gay and Alex Aiken (CMU)

read by Shinya Kawanaka

概要

- RC という C の方言で、
  - region based memory management library をもち
  - 静的解析でそれの overhead を減らした
- ものを作った

Region based memory management

- malloc/free の用にグローバルなメモリから取ってくるのではなく、region という単位で取ってからそれを分け与えるようなメモリ管理機構
- 同じ region からメモリを取ってくれば、物理的に近くなるのでキャッシュヒットしやすくなったり、メモリ割り当てコストが下がったりする

Sample Code

```
struct rlist { struct rlist* next; struct finfo* data; } *r1, *last = NULL;

region r = newregion();

while (...) {
  r1 = ralloc(r, struct rlist);
  r1->data = ralloc(r, struct finfo); // and fill data.
  r1->next = last; last = r1;
}

deleteregion(r);
```

既存の Region based memory management の問題点

- traditional なものは unsafe
  - region を破棄したときに、その region 内を指しているポインタが dangling pointer に
- check するものも static checking なものばかり(当時)

RC: C with region library は C の方言で、region の関数(マクロ)と annotation を入れたもの

- region APIs
  - region newregion(void);
  - region newsubregion(region r);
  - void deleteregion(region r);
  - type* ralloc(region r, type);
  - type* rarrayalloc(region r, size_t n, type);
  - region regionof(void* x);
- region annotation
  - sameregion (同じ region を指す)
  - traditional (region 管理されないメモリを指す)
  - parentptr (subregion の親を指す)

region annotation の例をひとつ

- 次のコードでは、next は 自分の region と同じ region のメモリを指す、と宣言できる

```
struct rlist {
  struct rlist *sameregion next;
  struct finfo *sameregion data;
};
```

実装

- RC から C へのトランスレータとして実装
- region は、reference counts で管理
- region annotation の違反は dynamic に検出
  - 例： sameregion のチェック (*p = newval; の時)
    - if (newval && regionof(newval) != regionof(p)) abort(); というコードを埋め込む
- これだと重いので、型システムで静的検査も行い、動的チェックがいらないと分かったところは動的チェックを取り除く

region の型システム

- u@σ (types)
  - region σ にある u
- ∃ρ/δ.τ
  - 何かリージョン ρ があって、property δ を満たすようなものが存在する τ
- T (region expr.)
  - null が指す region

$\tau = \mu@\sigma \mid \exists\rho/\delta.\tau$ (types)
$\mu = \texttt{region} \mid T[\sigma_1, \ldots, \sigma_m]$ (base types)
$\sigma = \rho \mid R \mid \top$ (region expressions)
$\delta = \sigma \leq \sigma \mid \neg\delta \mid \delta \vee \delta \mid (\delta)$ (region properties)
$\texttt{struct}\ T[\rho_1, \ldots, \rho_m]\{field_1 : \tau_1, \ldots, field_n : \tau_n\}$ (structure declarations)

T: type names, ρ: abstract regions, R: region constants

Figure 4: Region type language

型の例

$$
\begin{array}{l}
\mathbf{struct}\ L[\rho]\ \{ \\
\quad v : \exists \rho'.\mathbf{region}@\rho', \\
\quad next : \exists \rho'' / \rho'' = \top \vee \rho'' = \rho.L[\rho'']@\rho'' \\
\} \\
x : L[\rho]@\rho
\end{array}
$$

- struct L は、リージョン ρ に格納される
- v は、region ρ’ に格納され、region を指す。ρ’ の条件は特にない
- next は、リージョン ρ” に格納され、L[ρ”] を指す。ρ” は、T (null が格納される region) もしくは、ρ と同じ
- x は、リージョン ρ に格納され、L[ρ] 型である

rlang (ここで使うプログラム)

- straightforward な命令型言語
- new はリージョン指定が入る
- chk δ で、region property の動的検査

program ::= fn*

fn ::= $f[\rho_1,\ldots,\rho_m]/\delta(x_1:\tau_1,\ldots,x_n:\tau_n):\tau,\delta'$
is $[\rho'_1,\ldots,\rho'_p]x'_1:\tau'_1,\ldots,x'_q:\tau'_q, s, x$

s ::= $s_1; s_2$
| **if** x s_1 s_2
| **while** x s
| $x_0 = x_1$
| $x_0 = f[\sigma_1,\ldots,\sigma_m](x_1,\ldots,x_n)$
| $x_0 = x_1.field$
| $x_1.field = x_2$
| $x_0 = \mathbf{null}$
| $x_0 = \mathbf{new}\ T[\sigma_1,\ldots,\sigma_m](x_1,\ldots,x_n)@x'$
| **chk** δ

Some predefined functions:
`newregion[]/true()` : $\exists\rho.$`region@`ρ, `true`
`newsubregion[`ρ`]/true()` : $\exists\rho'/\rho' \leq \rho.$`region@`$\rho'$, `true`
`deleteregion[`ρ`]/true(`r : `region@`ρ`)` : `region@`$\top$, `true`
`regionof_`$T[\rho,\rho_1,\ldots]$`/true(`$x : T[\rho_1,\ldots]@\rho$`)` : `region@`$\rho$, `true`

Figure 5: *rlang*, a simple imperative language with regions

型検査

- judgement: δ, L ⊢ s, δ’
- property δ は、s を実行後に δ’ になる
- L は、live abstract region set

後は、型チェック

- 易しいものをひとつだけ

$$\frac{\mathbf{fv}(\delta') \subseteq L}{\delta, L \vdash \mathbf{chk}\ \delta', \delta \wedge \delta'}\ (\mathbf{check})$$

型チェック全体

- 詳しくは論文を見てください

$$\frac{\delta, L_s \vdash s, \delta' \quad x : \tau \quad \delta' \Rightarrow \delta'' \quad \mathrm{fv}(\delta) \cup \mathrm{fv}(\delta'') \subseteq \{\rho_1, \ldots, \rho_m\} \quad x'_1, \ldots, x'_q \text{ are dead before } s}{\vdash f[\rho_1, \ldots, \rho_m]/\delta(x_1 : \tau_1, \ldots, x_n : \tau_n) : \tau, \delta'' \text{ is } [\rho'_1, \ldots, \rho'_p]x'_1 : \tau'_1, \ldots, x'_q : \tau'_q, s, x} \text{ (fndef)}$$

$$\frac{x_0 : \tau_0 \quad x_1 : \tau_1 \quad \delta, L \vdash \tau_0 \leftarrow \tau_1, \delta', L'}{\delta, L \vdash x_0 = x_1, \delta'} \text{ (assign)}$$

$$\frac{x_0 : \tau_0 \quad x_1 : \mu_1 @ \sigma_1 \quad x_1.\mathit{field} : \tau'_1 \quad \delta \wedge \sigma_1 \neq \top, L \vdash \tau_0 \leftarrow \tau'_1, \delta', L'}{\delta, L \vdash x_0 = x_1.\mathit{field}, \delta'} \text{ (read)}$$

$$\frac{x_1 : \mu_1 @ \sigma_1 \quad x_1.\mathit{field} : \tau'_1 \quad x_2 : \tau_2 \quad \delta \wedge \sigma_1 \neq \top, L \vdash \tau'_1 \leftarrow \tau_2, \delta', L'}{\delta, L \vdash x_1.\mathit{field} = x_2, \delta'} \text{ (write)}$$

$$\frac{\begin{array}{c}\text{struct } T[\rho_1, \ldots, \rho_m]\{\mathit{field}_1 : \tau'_1, \ldots, \mathit{field}_n : \tau'_n\} \\ x_i : \tau_i \qquad \delta_i, L_i \vdash \tau'_i[\sigma_1/\rho_1, \ldots, \sigma_m/\rho_m] \leftarrow \tau_i, \delta_{i+1}, L_{i+1} \\ x_0 : \tau_0 \quad x' : \mathtt{region} @ \sigma' \quad \delta_{n+1}, L_{n+1} \vdash \tau_0 \leftarrow T[\sigma_1, \ldots, \sigma_m] @ \sigma', \delta', L'\end{array}}{\delta_1, L_1 \vdash x_0 = \mathtt{new}\ T[\sigma_1, \ldots, \sigma_m](x_1, \ldots, x_n) @ x', \delta'} \text{ (new)}$$

$$\frac{x_0 : \mu_0 @ \sigma_0 \quad \delta, L \vdash \mu_0 @ \sigma_0 \leftarrow \mu_0 @ \top, \delta', L'}{\delta, L \vdash x_0 = \mathtt{null}, \delta'} \text{ (null)} \qquad \frac{\mathrm{fv}(\delta') \subseteq L}{\delta, L \vdash \mathtt{chk}\ \delta', \delta \wedge \delta'} \text{ (check)}$$

$$\frac{\delta, L \vdash s_1, \delta' \quad \delta', L_{s_2} \vdash s_2, \delta''}{\delta, L \vdash s_1; s_2, \delta''} \qquad \frac{\delta, L_{s_1} \vdash s_1, \delta' \quad \delta, L_{s_2} \vdash s_2, \delta''}{\delta, L \vdash \mathtt{if}\ x\ s_1\ s_2, \delta' \vee \delta''} \qquad \frac{\delta \vee \delta'', L_s \vdash s, \delta''}{\delta, L \vdash \mathtt{while}\ x\ s, \delta \vee \delta''}$$

$$\frac{\begin{array}{c}f[\rho_1, \ldots, \rho_m]/\delta'(y_1 : \tau'_1, \ldots y_n : \tau'_n) : \tau', \delta'' \\ x_i : \tau_i \quad \delta_i, L_i \vdash \tau'_i[\sigma_1/\rho_1, \ldots, \sigma_m/\rho_m] \leftarrow \tau_i, \delta_{i+1}, L_{i+1} \quad \delta_{n+1} \Rightarrow \delta'[\sigma_1/\rho_1, \ldots, \sigma_m/\rho_m] \\ \delta_{n+1} \wedge \delta''[\sigma_1/\rho_1, \ldots, \sigma_m/\rho_m], L_{n+1} \vdash \tau_0 \leftarrow \tau'[\sigma_1/\rho_1, \ldots, \sigma_m/\rho_m], \delta''', L'\end{array}}{\delta_1, L_1 \vdash x_0 = f[\sigma_1, \ldots, \sigma_m](x_1, \ldots, x_n), \delta'''} \text{ (fncall)}$$

Assignment

$$\frac{\sigma' \in L \cup C_R \qquad \mathrm{fv}(\delta'[\sigma'/\rho]) \subseteq L \qquad \delta \Rightarrow \delta'[\sigma'/\rho] \quad \delta, L \vdash \tau[\sigma'/\rho] \leftarrow \tau', \delta'', L'}{\delta, L \vdash \exists \rho/\delta'.\tau \leftarrow \tau', \delta'', L'} \text{ (}\exists\text{gen.)}$$

$$\frac{\rho \notin L \qquad \delta \Rightarrow \delta'' \qquad \mathrm{fv}(\delta'') \subseteq L \qquad \delta'' \wedge \delta'[\rho/\rho'], L \cup \{\rho\} \vdash \tau \leftarrow \tau'[\rho/\rho'], \delta''', L'}{\delta, L \vdash \tau \leftarrow \exists \rho'/\delta'.\tau', \delta''', L'} \text{ (}\exists\text{inst.)}$$

$$\frac{\delta, L \vdash \sigma \leftarrow \sigma', \delta', L'}{\delta, L \vdash \mathtt{region} @ \sigma \leftarrow \mathtt{region} @ \sigma', \delta', L'} \qquad \frac{\delta, L \vdash \sigma \leftarrow \sigma', \delta_1, L_1 \qquad \delta_i, L_i \vdash \sigma_i \leftarrow \sigma'_i, \delta_{i+1}, L_{i+1}}{\delta, L \vdash T[\sigma_1, \ldots, \sigma_m] @ \sigma \leftarrow T[\sigma'_1, \ldots, \sigma'_m] @ \sigma', \delta_{m+1}, L_{m+1}}$$

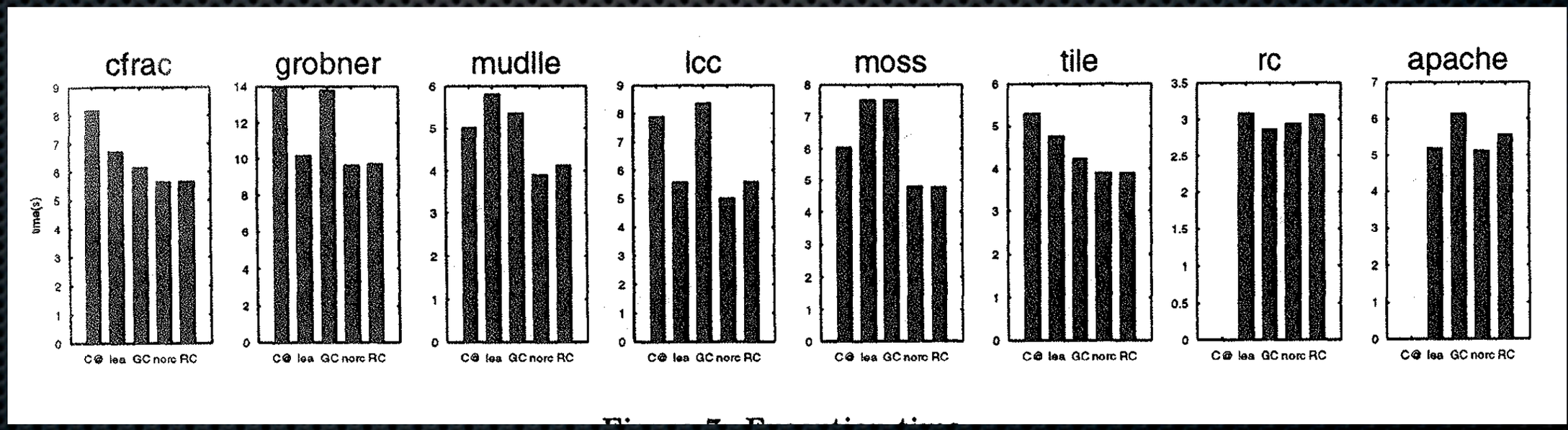
$$\frac{\sigma \in L \cup C_R \quad \delta \Rightarrow \sigma = \sigma'}{\delta, L \vdash \sigma \leftarrow \sigma', \delta, L} \qquad \frac{\rho \notin L \qquad \delta \Rightarrow \delta' \qquad \mathrm{fv}(\delta') \subseteq L}{\delta, L \vdash \rho \leftarrow \sigma', \delta' \wedge \rho = \sigma', L \cup \{\rho\}}$$

Figure 6: Region Type Checking

Benchmark 結果

RC は速い：malloc/free の -7% ~ 53%

- ５つ棒は左から、C@ (著者らの前システム), lea (Doug Lea の malloc/free 再実装), GC (Boehm-GC), norc (RC で reference counting を disable したもの), RC
- norc は RC の refcount がない版なので当然速い



静的検査で減った dynamic check 割合

- 灰色が safe assignment (dynamic check なし)
- 黒が dynamic check

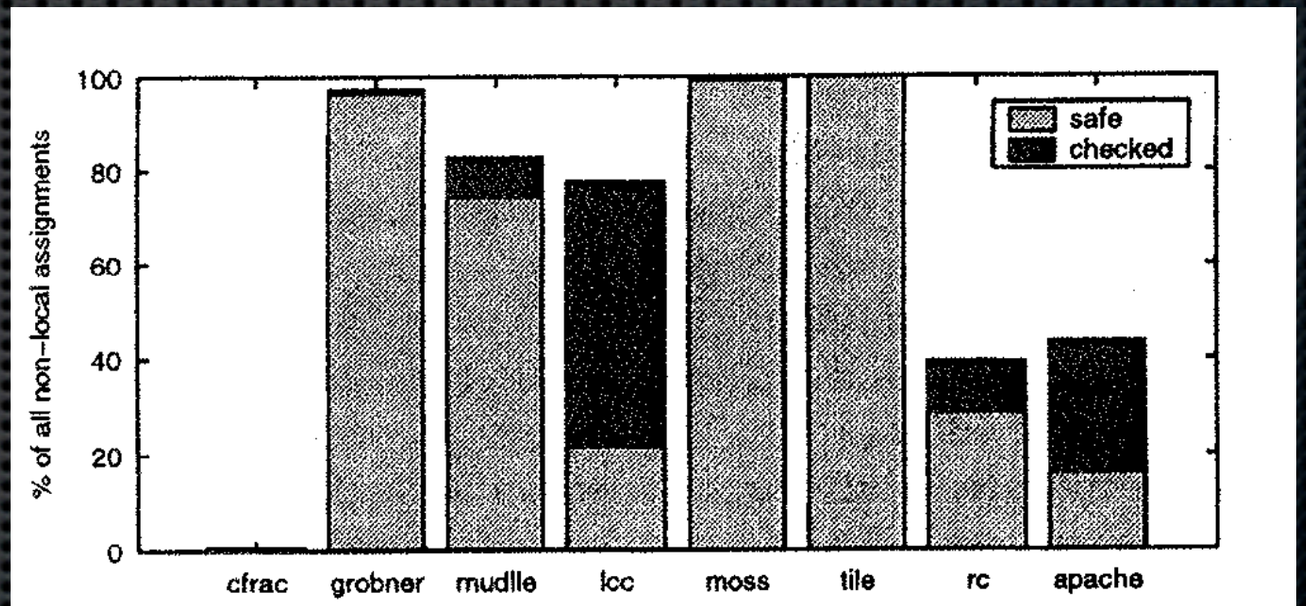


Figure 9: Details of reference count operations

まとめ

- region library があり、region annotation がある C 言語の方言 RC を作った
- malloc/free system に比べて 7% 遅い〜58% 速い
- 静的検査を行い、dynamic check を 21% - 99.99% 減らした

Citation Count: 81

A Framework for Reducing the Cost of Instrumented Code

Matthew Arnold and Barbara G. Ryder (IBM)

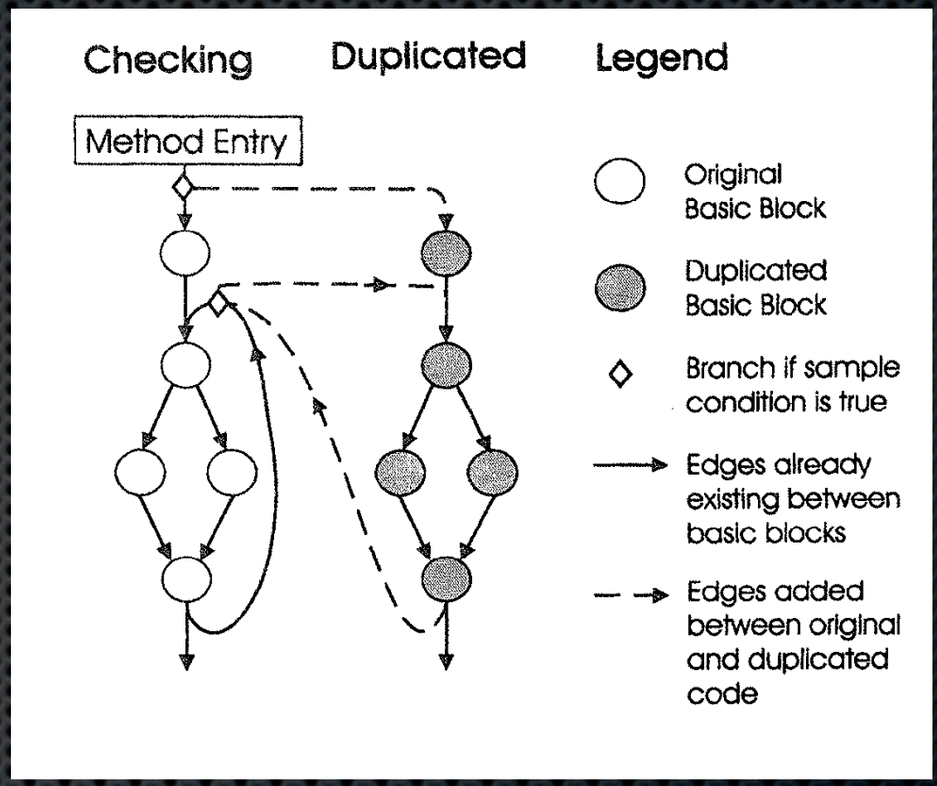
read by Shinya Kawanaka

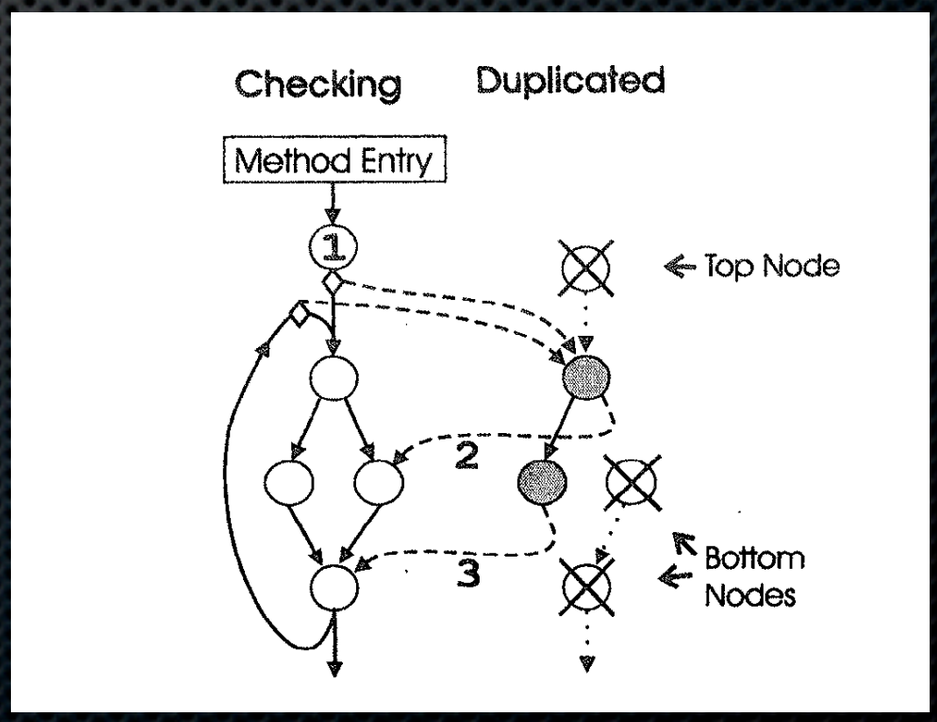
profile をなるべく軽く取りたい

- Instrumented Code (profile を取るようなコードが埋め込まれたコード) は、オーバーヘッドがあるので遅い
- なるべくそのようなコードを通らずに、そのコードを通ったときと同じような profile の結果を得たい

original と instrumented の両方を使う

- 元のコードから 何回かに一度 Instrumented Code へジャンプ
- I.C. からは必ず元のコードに戻る
- 部分的なコピーでも可





1000 回に 1 回 Instrumented Code を通れば、毎回通るのと同じような結果に

- 右は javac の図で、93 - 98 % 程度一致
- オーバーヘッドは1000回でわずか 6.3%