MITSUBISHI
三菱電機

Changes for the Better

MIPS PreLinkの適用と評価

三菱電機株式会社
八木孝介

概要

- MIPS向けPreLinkerおよび対応コンパイラ、ライブラリを入手し、ターゲットボード上で動作させた。
- ターゲットボード上でアプリケーション起動時間の計測を行いPreLinkの効果を確認した。

経緯

- 2005年6月CELF International Technical Conference in Yokohama JapanでPreLink技術の有効性が紹介される。
- 2005年11月Tokyo Technical JamboreeにてMIPSアーキテクチャでのPreLinkに関して質問
- 2006年4月ELC(worldwide Embedded Linux Conference)にてMIPS-SIG開催、MIPS PreLink実装の要望
- 2006年7月OLS(Ottawa Linux Symposium)時にMIPS社から開発着手の報告
- 2006年10月Tokyo Technical JamboreeにてMIPS社/CodeSourcery社から開発完了の報告

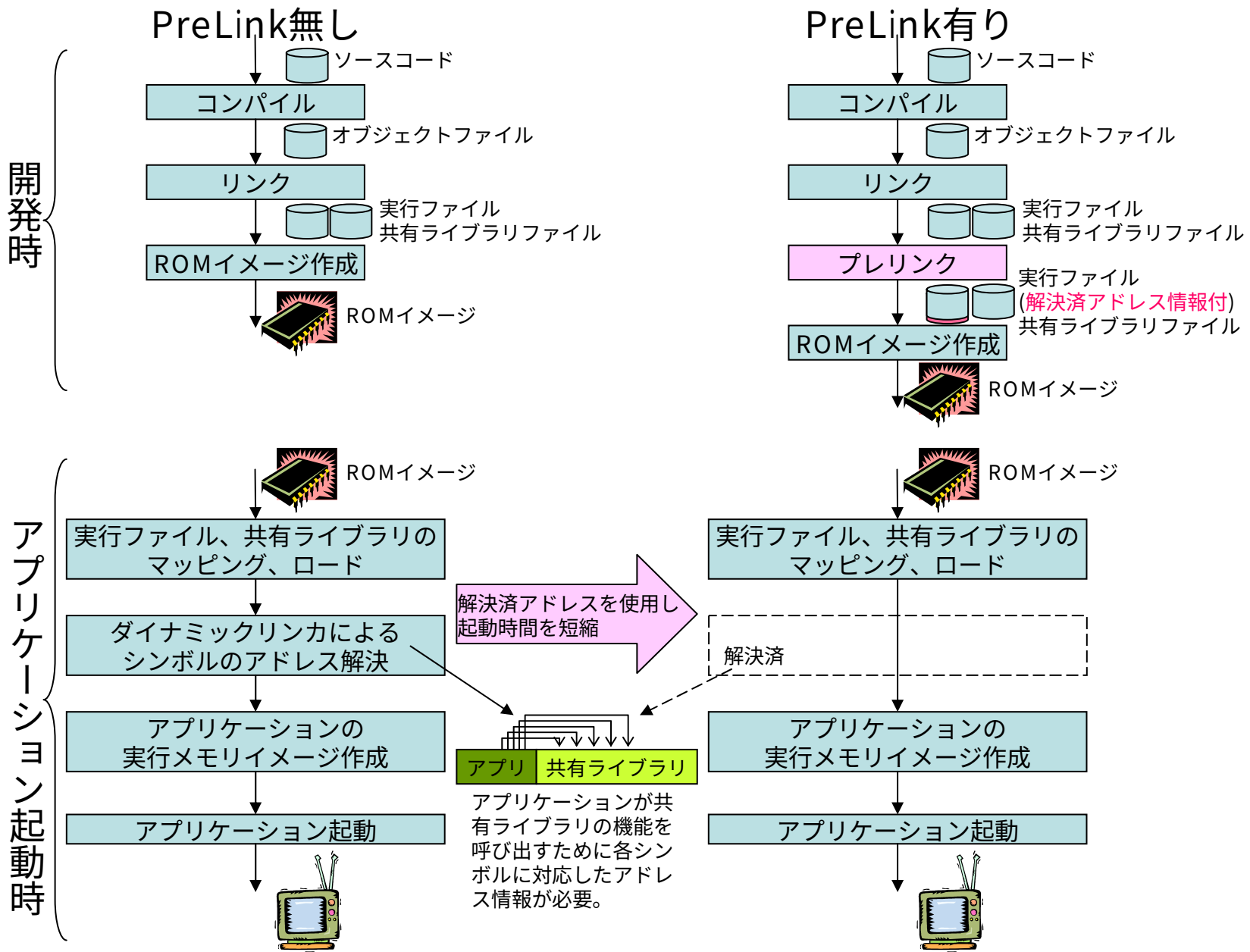
PreLinkとは

- 動的リンクタイプの共有オブジェクト（ライブラリなど）を使用するプログラムの起動を高速化する技術。
- 通常はプログラム実行時に動的に解決されるリンク情報を、PreLinkの場合はプログラム作成時にあらかじめ計算しておくことによってプログラム実行時の計算を省略し、起動高速化を可能とする。
- 静的リンクと動的リンクの中間的な性格を持つ。

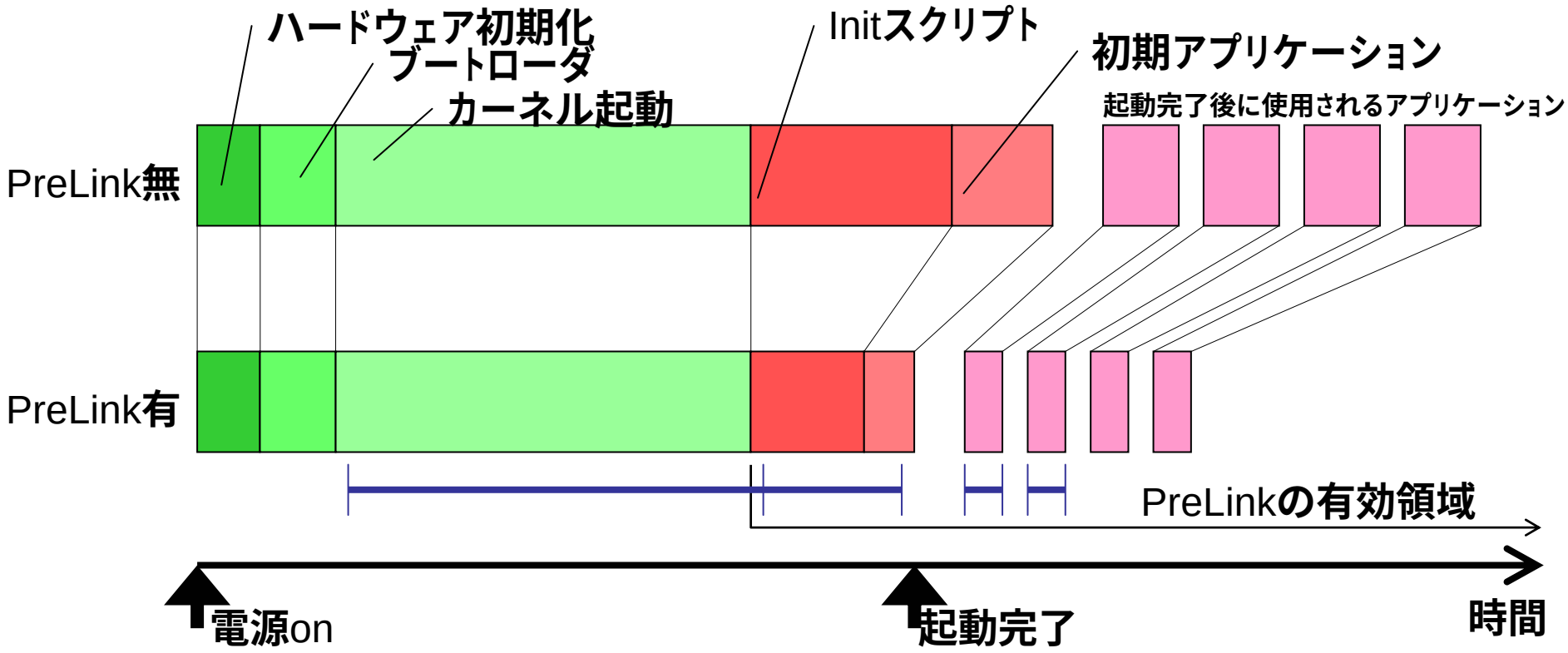
MITSUBISHI 三菱電機

PreLinkとは

Changes for the Better



PreLinkとは


ハードウェア初期化
ブートローダ
カーネル起動
Initスクリプト
初期アプリケーション
起動完了後に使用されるアプリケーション
PreLink無
PreLink有
PreLinkの有効領域
電源on
起動完了
時間

MIPS PreLink環境の作成

- **パッチの入手**
  - prelink.tar.gz (svn://sourceware.org/svn/prelink)
  - glibc-2.3.6-prelink.patch (sourceware.org/glibc)
  - binutils-2.16.1-prelink.patch (sourceware.org)
- **パッチ適用の上**tool-chain**の再構築**
- prelink**コマンドのコンパイル (今回はターゲット上でのセルフ**Prelinker**として作成した)**
- **ターゲットイメージの作成**
- **ターゲット上で**prelink**の実行**

ターゲット環境

- CPU**コア**：MIPS32(4KEc)
- **クロック**：300MHz
- RAM：384MByte
- Flush：64MByte

評価方法1(C)

src-gen/main.c

```
#include "incgen.h"

int main( int argc, char **argv )
{
        return 0;
}
```

inc-gen/incgen.h

```
extern void func001();
extern void func002();
extern void func003();
     :
```

lib-gen/func001.c

```
#include "incgen.h"
void func001() {}
```

- 左記のソースコードからライブラリ関数が一つずつ独立した共有ライブラリファイルとなるようにコンパイル
- コンパイルにはGnu Auto toolsを使用
- シェルスクリプトで無限ループを作成。timeコマンドで繰り返し計測コマンドの起動終了を繰り返し、10回の平均を取る。

評価方法1(C++)

src-gen/main.cpp

```
#include "incgen.h"

int main( int argc, char **argv )
{
        return 0;
}
```

inc-gen/incgen.h

```
extern void func001();
extern void func002();
extern void func003();
     :
```

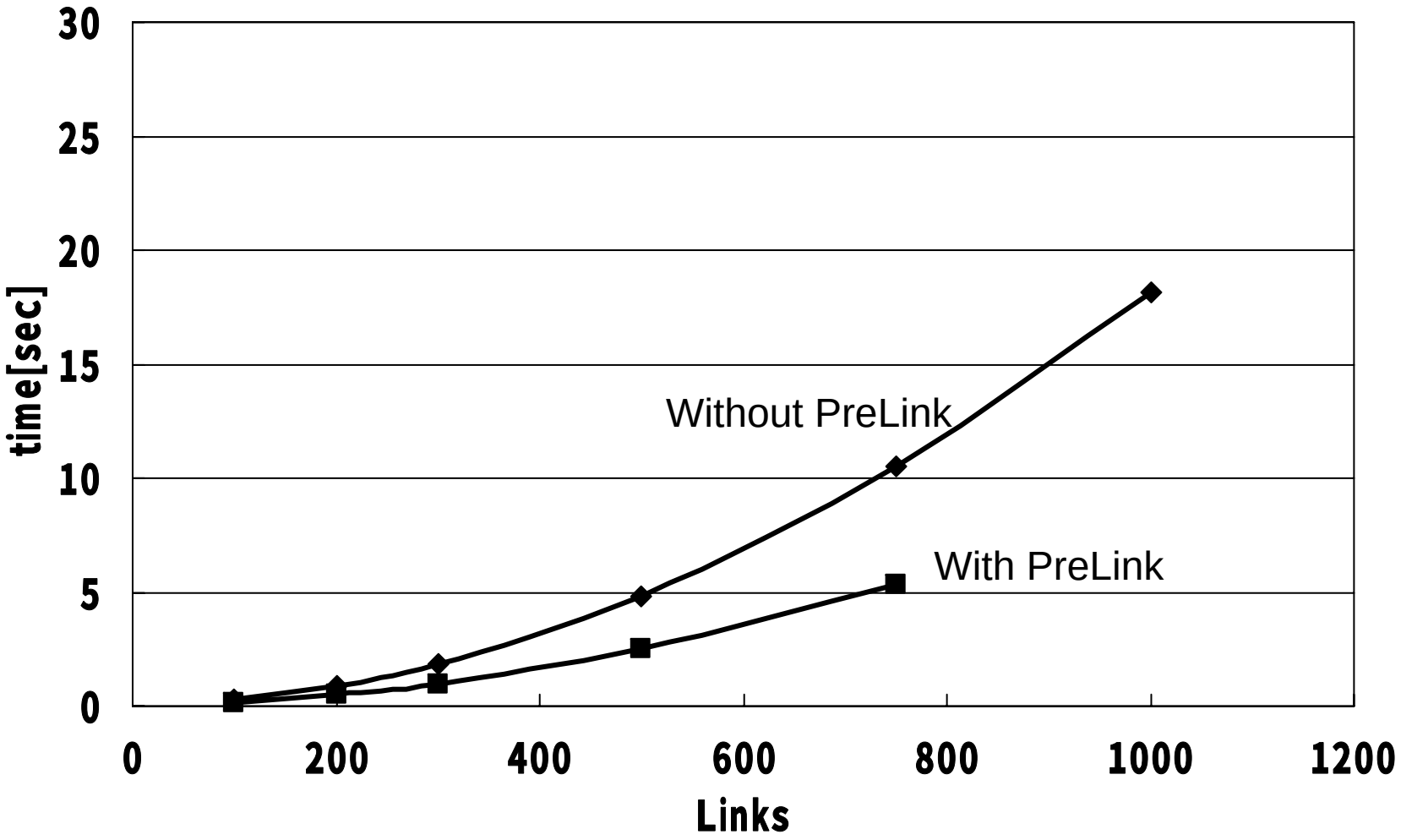
lib-gen/func001.cpp

```
#include "incgen.h"
void func001() {}
```

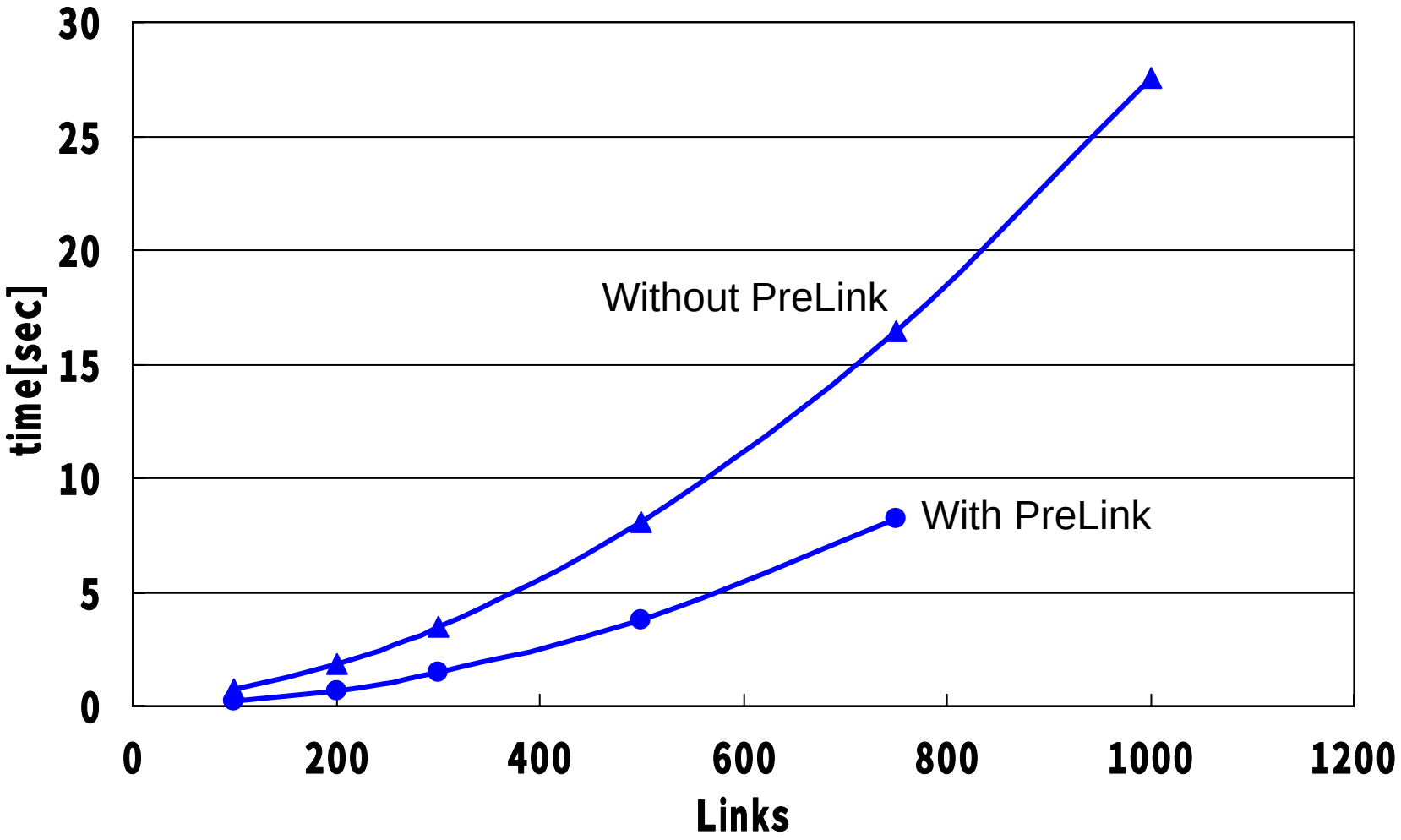
- 左記のソースコードからライブラリ関数が一つずつ独立した共有ライブラリファイルとなるようにコンパイル
- コンパイルにはGnu Auto toolsを使用
- シェルスクリプトで無限ループを作成。timeコマンドで繰り返し計測コマンドの起動終了を繰り返し、10回の平均を取る。

注：拡張子、コンパイラを除いてC版のテストプログラムと同一

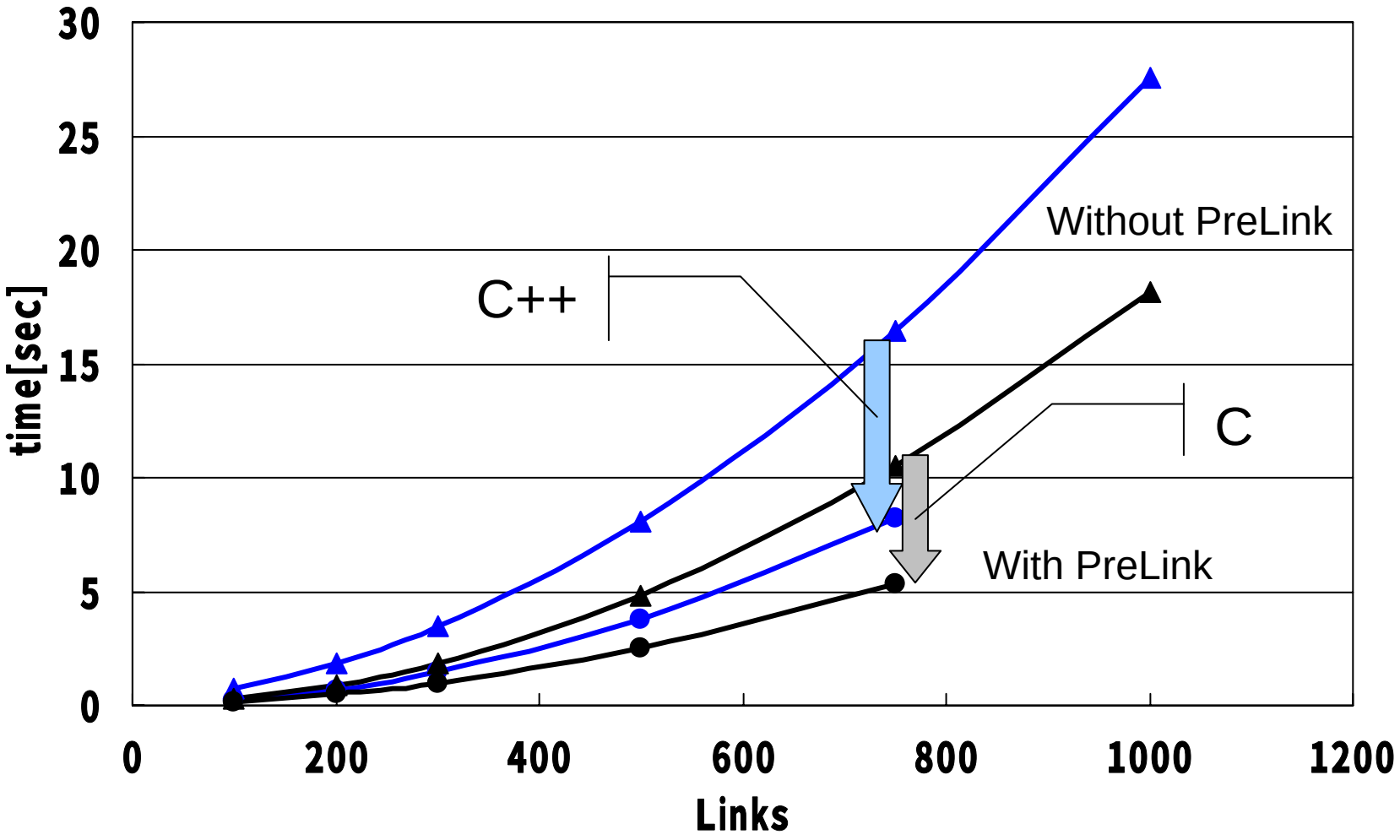
アプリケーション起動時間の計測(C)



アプリケーション起動時間の計測(C++)



アプリケーション起動時間の計測



システム起動時間計測

- initスクリプトの先頭、procfsマウント直後に/proc/uptimeの内容を一時ファイルに書き出す。
- initスクリプト完了後、最初に起動されるアプリケーション(with 300 dummy links)の先頭で/proc/uptimeの値を取得。保存しておいた値と比較してinitスクリプトから初期アプリケーションの起動までにかかる時間を計測
- 1.78secの起動時間短縮が確認された。

まとめ

- MIPS PreLinkの動作検証を行った。
- 200Linkの場合0.92s→0.49s(C)、1.88s→0.70s(C++)の結果が得られた。
- アプリケーション起動時間はリンク数に対して指数関数的に増加する。
- CよりもC++の方がアプリケーション起動時間が大きくPreLinkによる改善量も大きい。

Special Thanks

- MIPS Technologies
- CodeSourcery
- MIPS-SIG Members (CE-Linux Forum)