鲲鹏编译优化技术探索与软件优化实践

魏伟

HUAWEI

2021 InfoQ 技术大会近期会议推荐

—— 盘点一线大厂创新技术实践

北京站

GMTC 全球大前端技术大会

时间：2021年07月04-05日

地点：北京 · 国际会议中心

扫码查看完整日程

深圳站

ArchSummit 全球架构师峰会 th

时间：2021年07月23-24日

地点：深圳 · 大中华喜来登酒店

扫码查看大会专题

目录

HUAWEI

毕昇编译器介绍

毕昇编译器基于开源LLVM 10.0.1版本开发，并进行了优化和改进，同时将flang作为默认的Fortran语言前端编译器，是一种Linux下针对鲲鹏920的高性能编译器。除支持LLVM通用功能和优化外，做了以下增强：

- **高性能编译算法：**编译深度优化，增强多核并行化，自动矢量化等，大幅提升指令和数据吞吐量。
- **加速指令集：**结合NEON/SVE等内嵌指令技术，深度优化指令编译和运行时库，发挥鲲鹏架构最佳表现。
- **AI迭代调优：**内置AI自学习模型，自动优化编译配置，迭代提升程序性能，完成最优编译

支持鲲鹏微架构芯片及指令优化

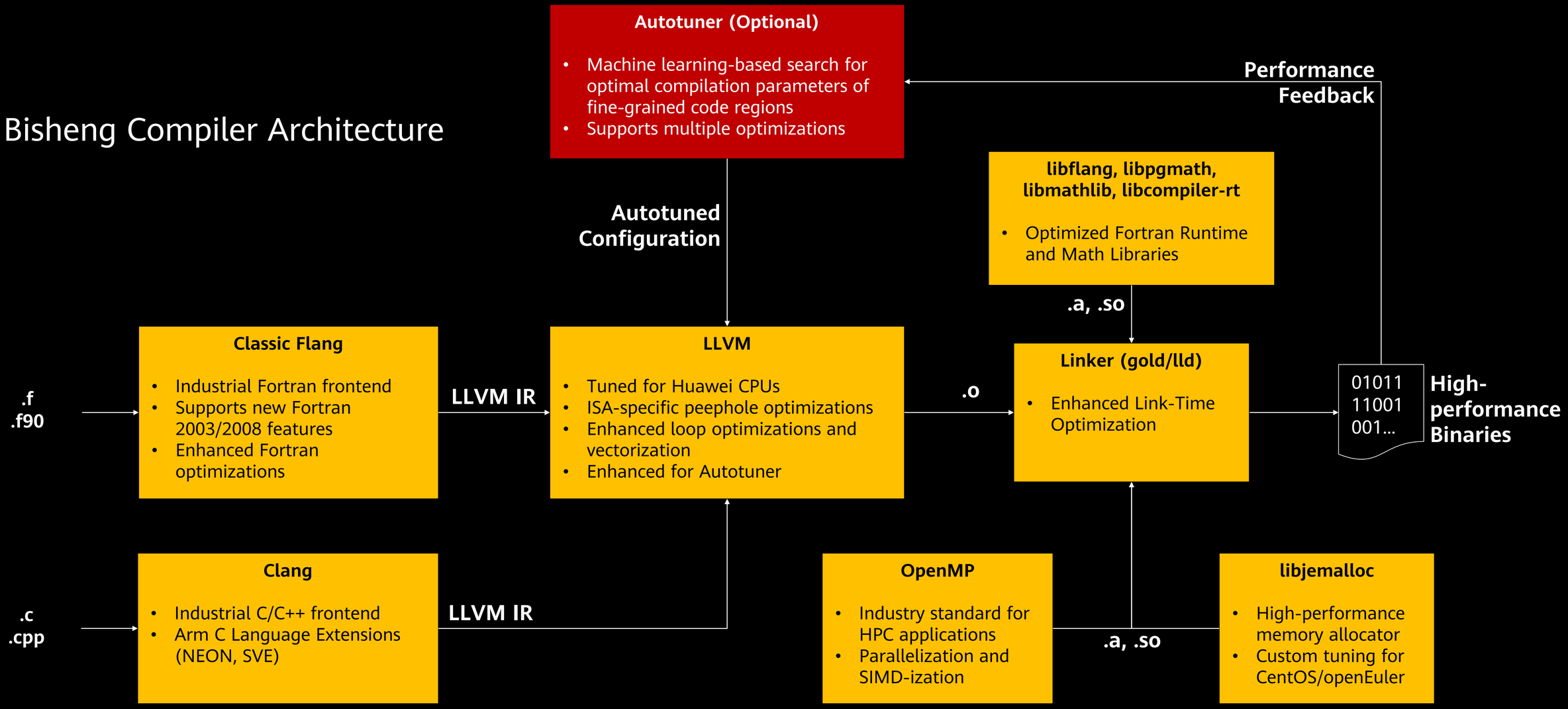
通过软硬协同提供相较开源LLVM更高的性能

集成Auto-tuner特性支持编译器自动调优

HUAWEI

毕昇编译器介绍：基本架构

- Bisheng Compiler Architecture



HUAWEI

目录

HUAWEI

毕昇编译器优化技术： 循环优化

循环(loop)优化是编译器中极为重要的一个优化手段，具有极为广泛及多样化的优化措施。编译器通过不同的优化方法来提高循环的性能。

提高缓存利用率

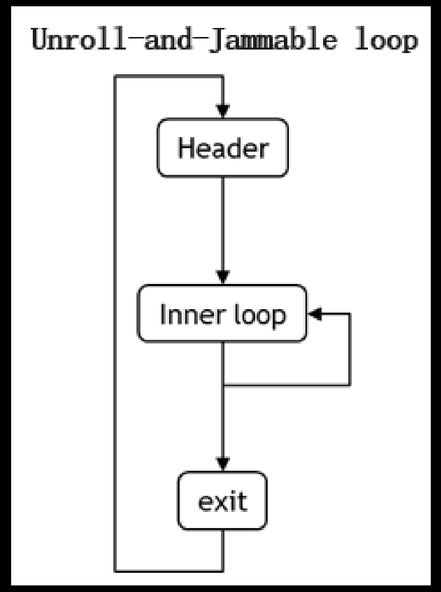
降低寄存器压力

减少动态指令数

复用不同迭代加载或计算值暴露其他优化的机会

> 向量化

> 指令调度



Loop Unroll and jam

对外层循环进行展开，并将内层循环进行合并：

改善内存和cache局部性及利用率

下图为简化版本，处理remainder loop部分没有标识在图中

```
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    a[j] += b[i][j] + b[i-1][j];
  }
}

for (int i = 0; i < n-n%2; i+= 2) {
  for (int j = 0; i < m; j++) {
    a[j] += b[i][j] + b[i-1][j];
    a[j] += b[i+1][j] + b[i][j];
  }
}
```

Loop Fusion/Distribution

这两个优化是互为相反的操作。Loop fusion会将两个循环的循环体尽可能的合成一个。Loop distribution则会将一个循环的循环体拆成两个独立的循环:

Loop fusion优化的作用:

- 直接复用其它循环中的值
- 暴露更多的指令调度机会

Loop distribution优化的作用:

- 减少循环中的寄存器压力
- 暴露更多的矢量化机会
- 给其他的循环优化提供更多的可能性

在下面的例子中，我们会将循环中的头两条语句放在一起，这样就能复用从b数组加载出来的值，但是会将第3条语句拆分出去，因为它存在跨迭代依赖，无法做循环矢量化。（需要注意的是，在做loop distribution之前，编译器有可能会对循环的头两条指令已经做了矢量化，比如SLP矢量化）

```
for (int i = m; i < n; i++) {
  a[i] = b[i-1] + b[i];
  c[i] = b[i-1] - b[i];
  d[i-1] = d[i-2] + 1;
}

for (int i = m; i < n; i++) {
  a[i] = b[i-1] + b[i];
  c[i] = b[i-1] - b[i];
}

for (int i = m; i < n; i++) {
  d[i-1] = d[i-2] + 1;
}
```

HUAWEI

毕昇编译器优化技术： 循环优化

Loop Unrolling

将循环体复制多遍，从而减少循环迭代次数：

- 可以减少动态的指令数量
- 发现更多的优化机会点，比如数据复用，范围更广的指令调度，以及提高矢量化的数据并发度

```
for (int i = 0; i < n; i++) {
  a[i] = b[i-1] + b[i];
}
```

```
int ub = n - n%2;
for (int i = 0; i < ub; i += 2) {
  a[i] = b[i-1] + b[i];
  a[i+1] = b[i] + b[i+1];
}
if (i < n)
  a[i] = b[i-1] + b[i];
```

Loop Unswitching

将循环不变量判断条件提出循环并进行循环复制，生成相应的版本：

- 帮助发现更多的优化机会点
- 减少分支跳转的执行次数

```
for (int i = 0; i < n; i++) {
  a[i] = 1;
  if (b[j] > 10)
    a[i] += 1;
}
```

```
if (b[j] > 10)
  for (int i = 0; i < n; i++)
    a[i] = 1;
else
  for (int i = 0; i < n; i++)
    a[i] = 2;
```

```
void f (std::map<int, int> m)
{
  for (auto it = m.begin (); it != m.end (); ++it) {
    /* if (b) is semi-invariant. */
    if (b) {
      b = do_something();    /* Has effect on b */
    } else {
      /* No effect on b */
    }
    statements;              /* Also no effect on b */
  }
}
```

Loop split

```
void f (std::map<int, int> m)
{
  for (auto it = m.begin (); it != m.end (); ++it) {
    if (b) {
      b = do_something();
    } else {
      ++it;
      statements;
      break;
    }
    statements;
  }
  for (; it != m.end (); ++it) {
    statements;
  }
}
```

```
void f (std::map<int, int> m)
{
  for (auto it = m.begin (); it != m.end (); ++it);
}
```

Loop delete

```
void f (std::map<int, int> m)
{
}
```

HUAWEI

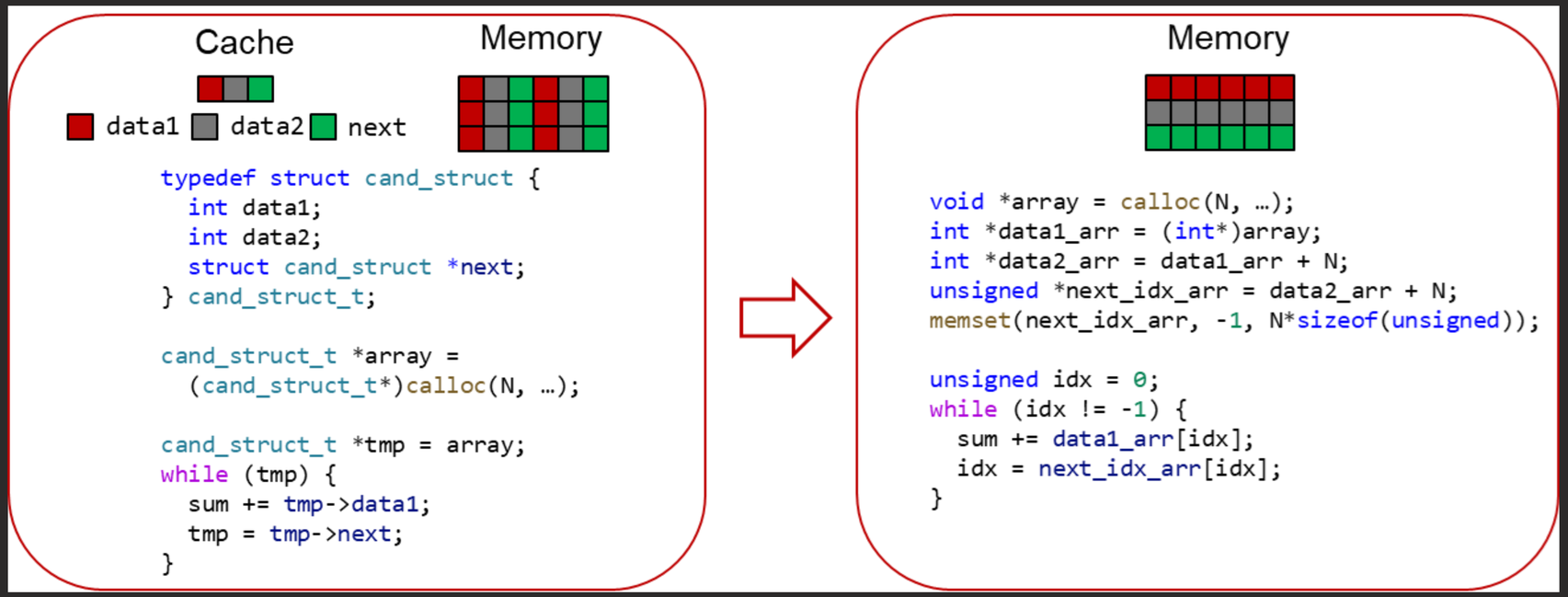
毕昇编译器优化技术： 结构体内存布局优化

- **结构体内存布局优化**

✓结构体内存布局优化基于全程序（Whole-program）优化，用以提高缓存（cache）利用率。

✓优化的主要手段是将结构体数组转换为数组结构体。

✓结构体可以是显式的，也可以通过检查循环中的数组使用情况来推断它们。

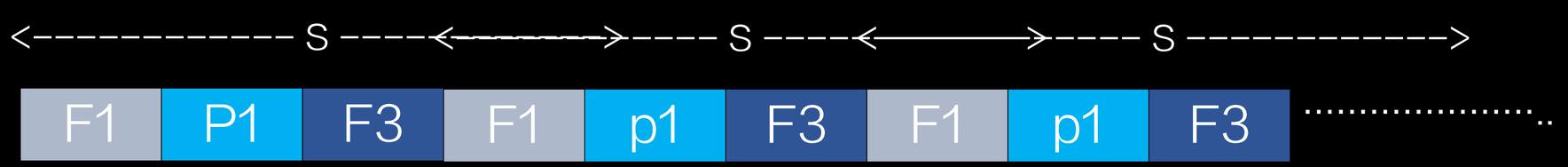


HUAWEI

毕昇编译器优化技术：结构体指针压缩优化

```
结构体 S
{
   Field F1;
   S * p1;
   Field F3;
}
```

针对 S[NUM]内存分配大小的变化




```
结构体 S_shrink
{
   Field F1;
   unsigned int ps;
   Field F3;
};

S_shrink *ps_head
```

变换分配为 S_shrink[NUM]内存大小

将指针成员由64bits压缩至32bits，

减小每个结构体node的内存体积。

被压缩的域成员指针外提为全局结构体指针ps_head，

ps变换为相对基址的偏移。

降低内存使用
提升d-cache
命中率

HUAWEI

毕昇编译器优化技术： 自动向量化

- 在并行计算中，自动矢量化是自动并行化的一个特殊场景，其中计算机程序从一次处理一对操作数的标量实现转换为矢量实现，它能在一次操作中同时处理多对操作数。比如，基于AArch64的鲲鹏920处理器，具有32个128 bits的矢量寄存器，可以一次操作4路32位或者2路64位的数据。
- 毕昇编译器重点优化了循环矢量化及SLP矢量化，充分保持程序局部性，高效提升计算密集型场景的性能。

✓ 循环向量化原理：

```
void foo(int *a, int *b,
         int *c, int n) {
  for (i=0; i<n; i++)
    a[i] = b[i] + c[i];
}
```

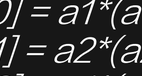
```
void foo(int *a, int *b,
         int *c, int n) {
  for (i=0; I < n/4; i+=4) {
    a[i+0] = b[i+0] + c[i+0];
    a[i+1] = b[i+1] + c[i+1];
    a[i+2] = b[i+2] + c[i+2];
    a[i+3] = b[i+3] + c[i+3];
  }
}
```

```
void foo(int *a, int *b,
         int *c, int n) {
  for (i=0; I < n/4; i+=4)
    a[0..3] = b[0..3] + c[0..3];
}
```

✓ SLP向量化原理：

(superword-level parallelism)

```
void foo(int a1, int a2,
         int b1, int b2,
         int *A) {
  A[0] = a1*(a1 + b1);
  A[1] = a2*(a2 + b2);
  A[2] = a1*(a1 + b1);
  A[3] = a2*(a2 + b2);
}
```



```
void foo(int a1, int a2,
         int b1, int b2,
         int *A) {
  A[0..3] = {a1,a2,a1,a2} * ({a1,a2,a1,a2} + {b1,b2,b1,b2});
}
```

HUAWEI

毕昇编译器优化技术：鲲鹏向量指令生成及优化

场景一：

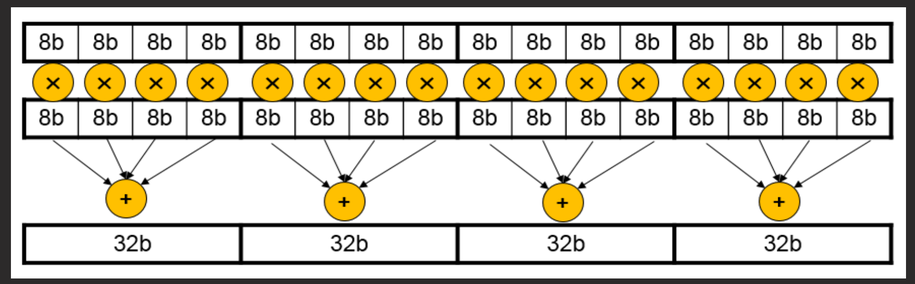
```
int sum = 0;  //pix1 and pix2 are uint8_t
for( int x = 0; x < 16; x++ )
  sum += abs( pix1[x] – pix2[x] );
```

```
movi   v1.16b,#1
ldr    q3, [x12, x0]
ldr    q2, [x13, x1]
uabd       v2.16b, v2.1       6b, v3.16b
udot       v0.4s, v2.16b, v1.16b
add    x8, x10, x0
add    x9, x11, x1
```

编译选项：

-sad-pattern-recognition=[true/false]

默认为true



场景二：

```
for( int x = 0; x < i_width; x++ )
  dst[x] = ( src1[x] + src2[x] + 1 ) >> 1;
```

```
ldr     d0, [x2, x16]
ldr     d1, [x4, x16]
ushll       v0.8h, v0.8b, #0
ushll       v1.8h, v1.8b, #0
mvn         v0.16b, v0.16b
sub         v0.8h, v1.8h, v0.8h
shrn        v0.8b, v0.8h, #1
str     d0, [x0, x16]
```

```
ldr     d0, [x2, x16]
ldr     d1, [x4, x16]
urhadd      v0.8b, v1.8b, v0.8b
str     d0, [x0, x16]
```

编译选项：

-aarch64-hadd-generation=[true/false]

默认为**true**

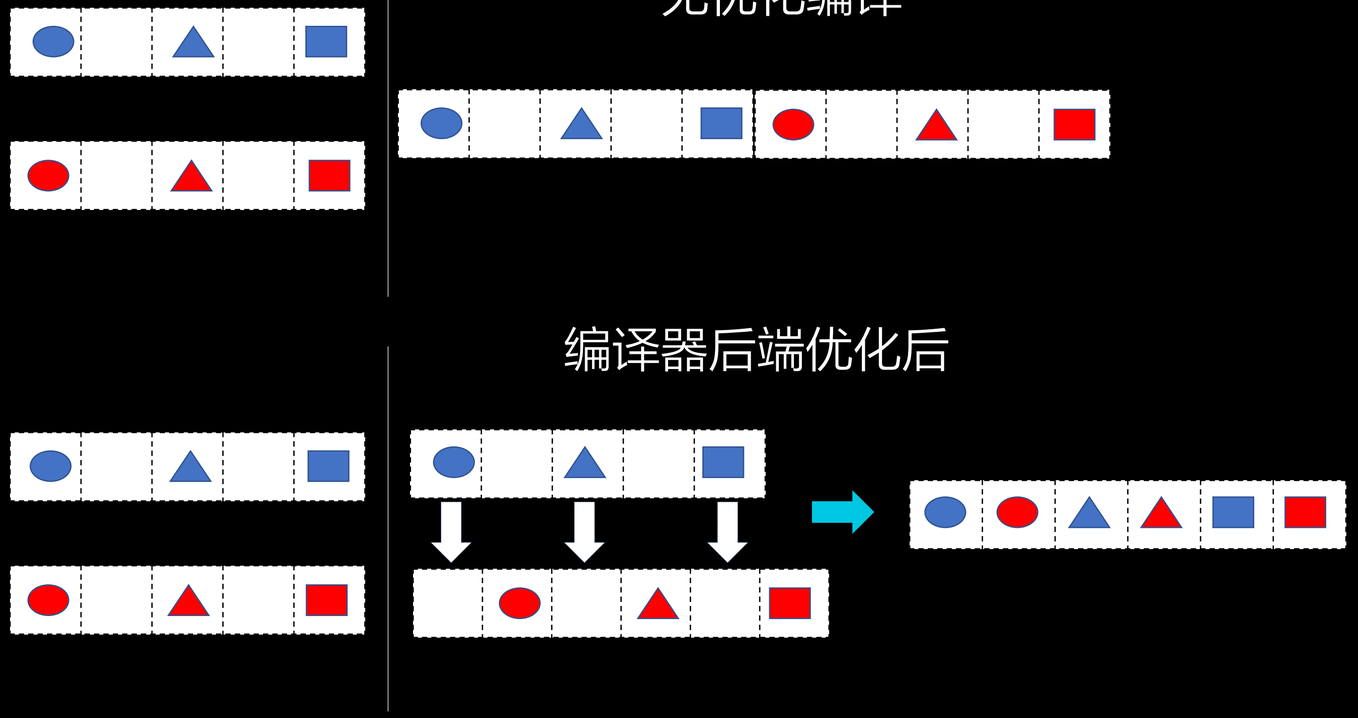
HUAWEI

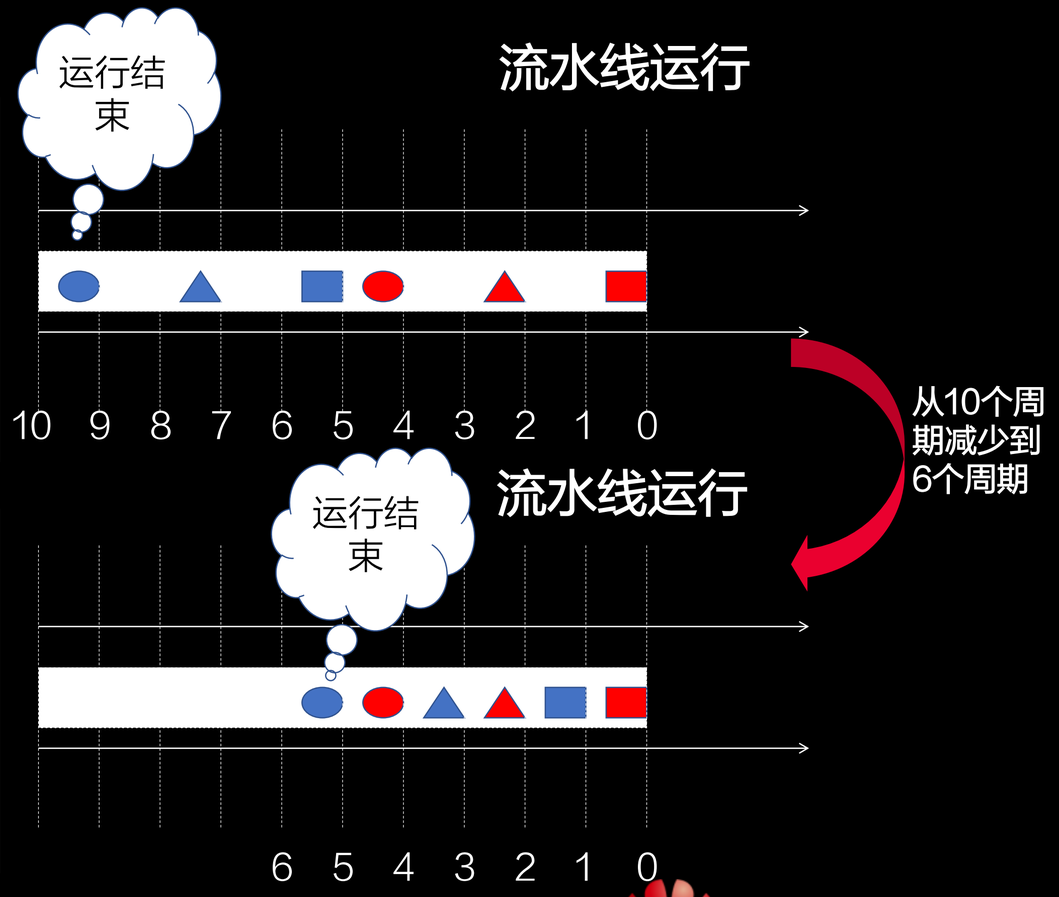
毕昇编译器优化技术：Pipeline优化，代码最佳执行效率

代码：

V1 = V2 + V3
V0 = V1 − V2
V5 = V0 * V2

K1 = K2 + K3
K0 = K1 − K2
K5 = K0 * K2

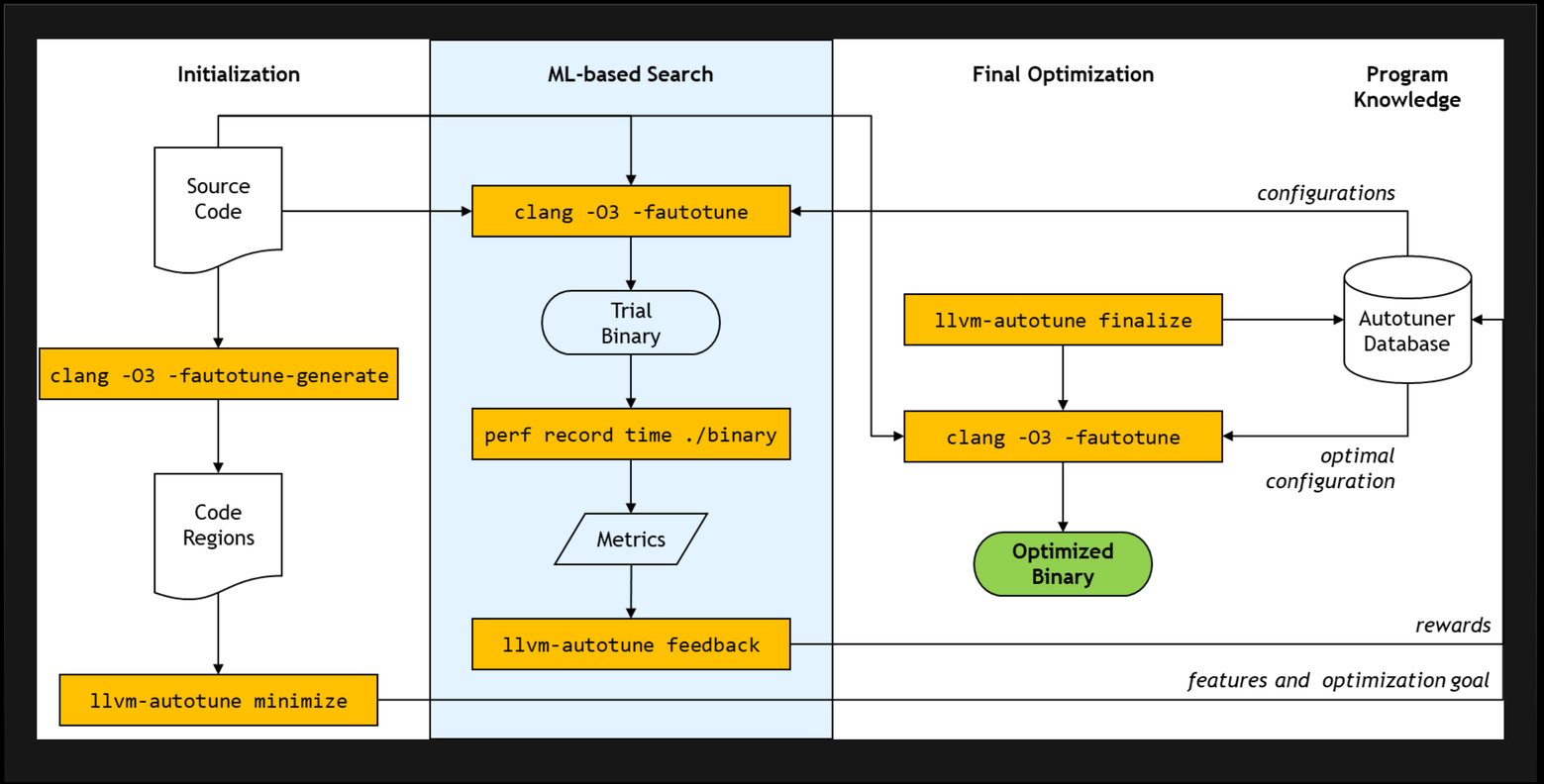




HUAWEI

毕昇编译器优化技术： Autotuner

- Autotuner基本架构



引入基于ML的自动搜索技术(ML-based Search)

关键技术点：

1、知识库(Autotuner Database)：根据静动态分析信息，建立知识库，支持决策系统进行优化；

2、优化决策系统(Optimal configuration)：根据热点和性能评估信息、知识库信息，综合考虑确定优化措施；

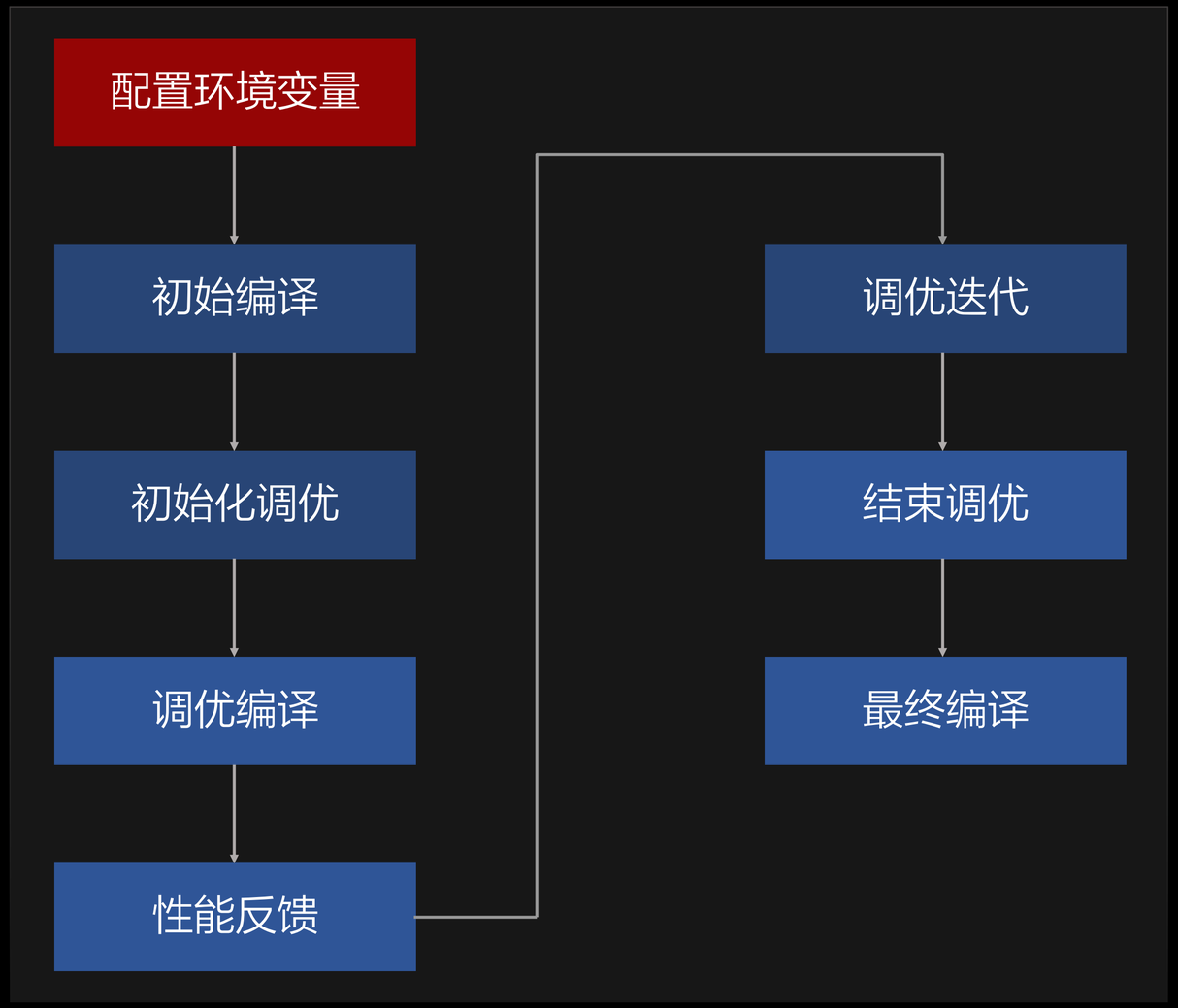
3、热点标记和性能评价：热点标记，瓶颈检测，性能评估；

4、查找驱动(Feedback)：将优化决策系统反馈的优化建议，反馈给编译器执行优化措施；

HUAWEI

毕昇编译器优化技术： Autotuner

Autotuner关键流程



Autotuner关键流程：

1、使用环境变量AUTOTUNE_DATADIR指定调优相关的数据的存放位置；

2、添加毕昇编译器选项-fautotune-generate， 完成初始编译，生成调优机会；

3、运行llvm-autotune命令，初始化调优任务。生成最初的编译配置供下一次编译使用；

4、添加毕昇编译器选项-fautotune，读取当前AUTOTUNE_DATADIR中的配置并编译；

5、运行程序，并根据自身需求获取性能数字，使用llvm-autotune feedback反馈。

6、根据用户设定的迭代次数，重复调优编译和性能反馈步骤进行调优迭代。

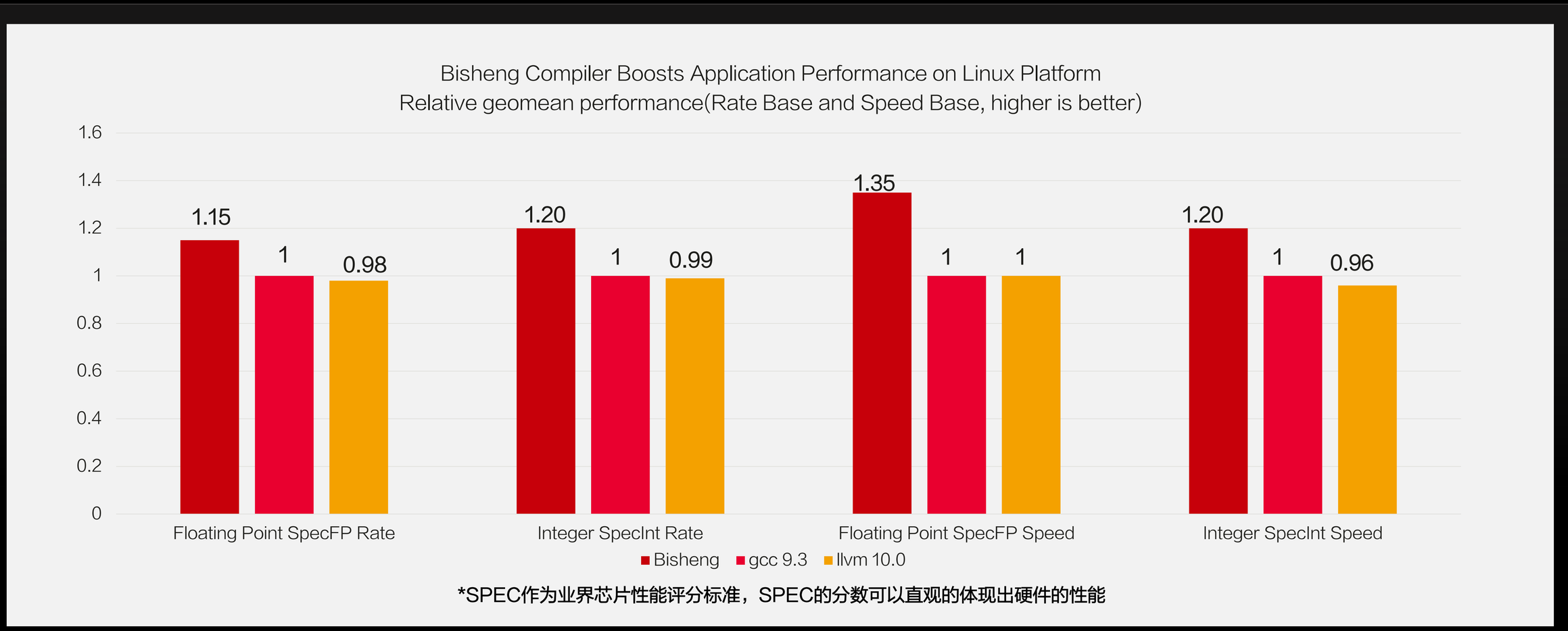
7、进行多次迭代后，可选择终止调优，并保存最优的配置文件。

8、使用上一步得到的最优配置文件，进行最后编译。

HUAWEI

毕昇编译器优化效果：业界CPU Benchmark跑分

毕昇编译器与鲲鹏芯片协同，通过编译器技术充分发挥芯片的性能，提升鲲鹏硬件平台上业务的性能体验。**基于鲲鹏上编译器优化，SPEC2017性能比gcc平均高20%以上。**

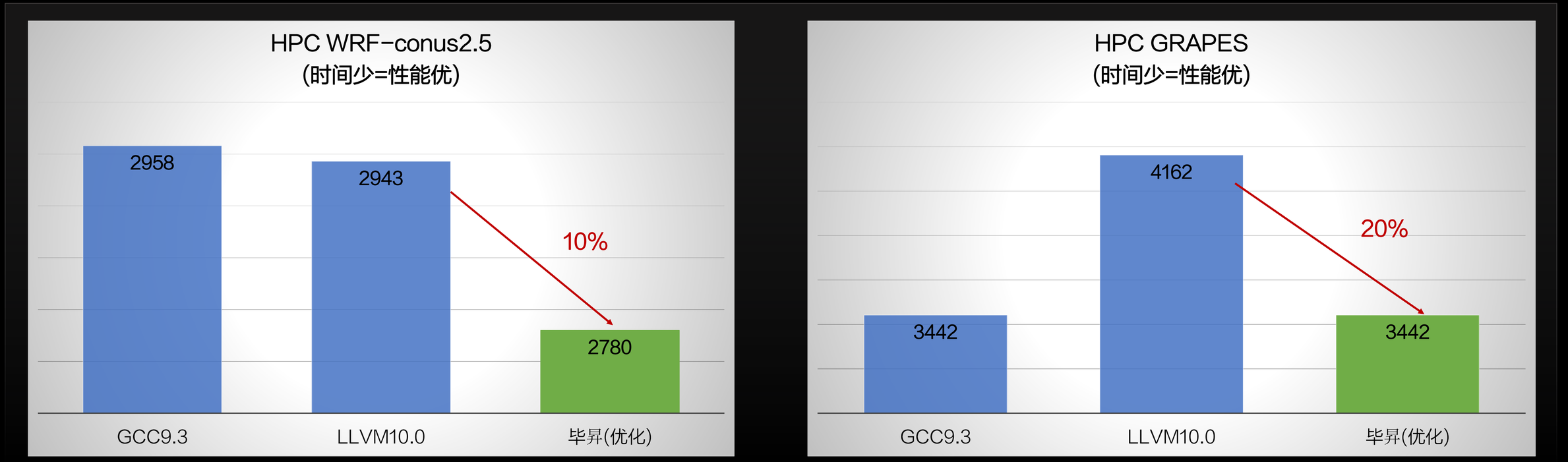


*SPEC作为业界芯片性能评分标准，SPEC的分数可以直观的体现出硬件的性能

HUAWEI

毕昇编译器优化效果：HPC典型应用性能提升

- HPC场景典型应用优化，结合鲲鹏芯片特性，通过毕昇编译器优化技术提升HPC 应用的性能体验。



HUAWEI

目录

HUAWEI

毕昇编译器：Fortran语言及特性支持

- **毕昇编译器flang语言前端基于Classic Flang构建，具有如下基本特性：**

✓ 支持F2003、F2008（Coarray特性除外）语言标准；

✓ 最高支持四倍浮点精度（real16）数据类型，15维数组数据类型；

✓ 支持OpenMP4.0、OpenMP4.5并行化编程模型；

✓ 支持DWARF4标准；

✓ 支持IEEE-754浮点类型标准；

✓ 多种Pragma引导语支持，如软件prefetch，omp simd，unroll，vector等

!$mem prefetch

内存引用方面的引导语，指示编译器将特定数据从main memory加载到L1/L2 cache。

```
!$mem prefetch array1, array2, array2(i + 4)
DO i=1,100
   array1(i – 1) = array2(i – 1) + array2(i + 1)
END DO
```

!dir$ ivdep

指示编译器忽略迭代循环的内存依赖性，并进行向量化。

```
!dir$ ivdep
DO i=1, ub
      array1(i) = array1(i) + array2(i)
END DO
```

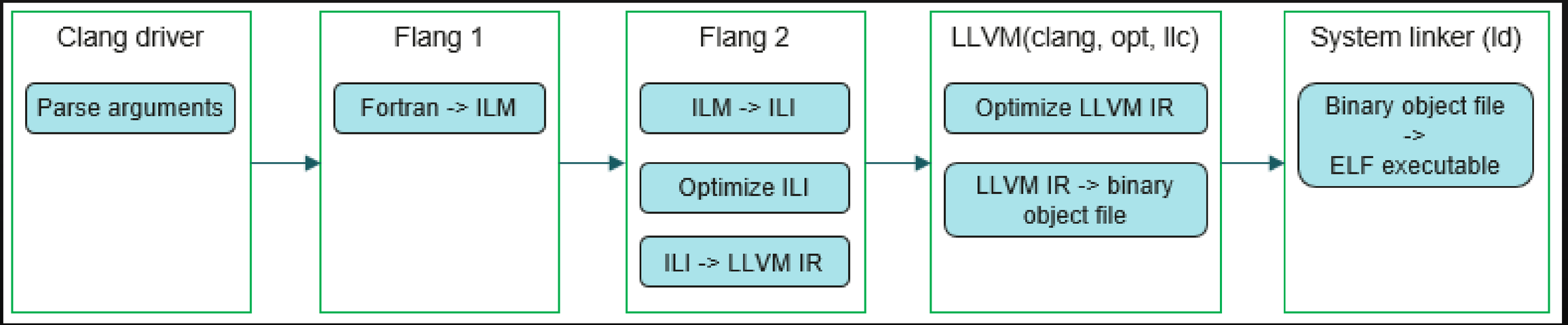
HUAWEI

Fortran语言及特性支持

- Classic Flang支持了部分F2008特性。
- 毕昇Flang编译器实现了如下语言特性支持，基本完成支持了F2008语言标准（Coarray特性暂不支持）。

特性类型	特性描述
数据声明	Quad-precision FP Math
	Maximum rank + corank = 15
	Allocatable components of recursive type
	Implied-shape array
	Type statement for intrinsic types
	Declaring type-bound procedures
	Value attribute is permitted for any nonallocatable nonpointer noncoarray
	In a pure procedure the intent of an argument need not be specified if it has the value attribute
数据对象访问	Omitting an ALLOCATABLE component in a structure constructor
	Pointer function reference is a variable
位操作内建函数	Bit sequence comparison
	Combined shifting
	Masking bits
	Shifting bits
	Merging bits
	Bit transformational functions
内建函数	Euclidean vector norms
	Optional argument RADIX added to SELECTED REAL KIND
	Parity function
程序和过程	Null pointer or unallocated allocatable as absent dummy arg.
	Non pointer actual for pointer dummy argument
	Generic resolution by procedureness
	Generic resolution by pointer vs. allocatable
Fortran 77特性	Variable FORMAT expressions

毕昇编译器：Flang 编译架构



- **Flang1阶段：**
  - ✓ 扫描Fortran代码并生产token；
  - ✓ 解析token生成语法树和符号表；
  - ✓ 生成标准语法树；
  - ✓ 将标准语法树生成ILM（Flang定义的一种中间语言助记符）。

- **Flang2阶段：**
  - ✓ 将ILM展开到ILI（Flang定义的中间语言指令）；
  - ✓ 通过编译优化算法生成优化后的ILI；
  - ✓ 根据LLVM IR语法生成LLVM IR；

HUAWEI

毕昇Flang 编译器特性：未初始化浮点变量检测

如下面这个程序：

```
 1 program test
 2   real :: uninita(2)
 3   real :: inita(2) = (/2, 5/)
 4
 5   call add_arr(uninita, inita)
 6
 7 contains
 8
 9 subroutine add_arr(arr1, arr2)
10 real :: arr1(:)
11 real :: arr2(:)
12 real :: rlt(2)
13 rlt = arr2 + arr1
14
15 print *, rlt
16
17 end subroutine add_arr
18 end program test
```

- ✓ 在变量初始化或定值前引用其值可能会导致应用程序错误，这些错误可能是随机运行错误，并且难以调试，特别是作为过程参数传递的变量。
- ✓ 毕昇Fortran编译器提供一种检测方法，通过将此类浮点数据初始化为NaN值，然后捕获该数据是否在浮点操作中使用，如果使用则会发生的浮点无效异常，从而达到检测为初始化浮点变量检测目的。
- ✓ 可以看出uninita数组未初始化，子例程中arr1的值是随机值，经过第13行的计算后，rlt值也是未定义的，可能导致运行问题。
- ✓ 用户可以通过以下命令将uninita数组元素的值自动为NaN。

  flang -init=nan - fcheck=uninit-fp test.f90
- ✓ 这时，再运行该例程，第十三行将会出现NaN数据参与计算，运行时将会报告一个异常错误提示，用户可以看到以下提示：

  These is possible uninitialized real/complex variable.

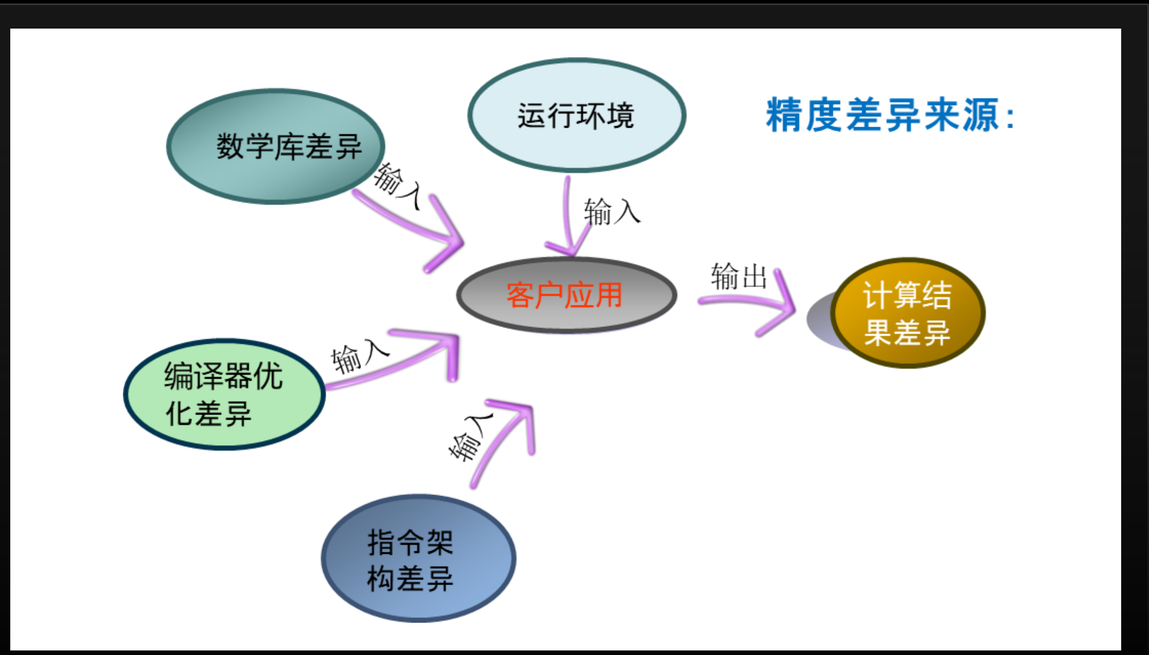
HUAWEI

目录

HUAWEI

毕昇编译器：浮点精度调优

HPC应用（如气象WRF，Grapes等）迁移至鲲鹏服务器后，往往存在计算结果与原有（以往）数据结果比对不上的场景。精度的差异主要来自四个方面：数学库、编译器优化、架构指令差异、运行环境。



对应解决方案

- 编译器优化差异：1.支持fp-model等精度控制选项 2.编译器优化行为调整及精度优化
- 指令架构差异：结合转码工具分析指令差异，使用软浮点模拟等手段消除差异
- 数学库差异：改写数学库函数，或使用不同精度数学库
- 运行环境差异：集群和多核环境差异(如MPI及OpenMP等)

HUAWEI

毕昇编译器：浮点精度调优

- 毕昇编译器浮点精度选项支持（-ffp-model/-ffast-math）

-ffast-math	-fno-honor-infinities	Assume no infinities.
	-fno-honor-nans	Assume no NaNs.
	-fno-math-errno	Disable setting the errno variable as required by C89/C99 when calling math library routines; default for gfortran.
	-ffinite-math-only	Allow floating-point optimizations that assume arguments and results are not NaNs or +-Inf.
	-fassociative-math	Allow floating point operations to be reassociated.
	-freciprocal-math	Allow division operations to be transformed into multiplication by a reciprocal. This can be significantly faster than an ordinary division but can also have significantly less precision.
	-fno-signed-zeros	Allow optimizations that ignore the sign of floating point zeros.
	-fno-trapping-math	Allow optimizations that assume that floating point operations cannot generate traps such as divide-by-zero, overflow and underflow.
	-ffp-contract=fast	Generate fused multiply-and-add instructions. Default is "fast".

-ffp-model=	precise	-ffp-contract=fast -fno-rounding-math
	strict	-ftrapping-math -frounding-math -ffp-exception-behavior=strict
	fast	-menable-no-infs -menable-no-nans -menable-unsafe-fp-math -fno-signed-zeros -mreassociate -freciprocal-math -ffp-contract=fast -fno-rounding-math -ffast-math -ffinite-math-only

HUAWEI

毕昇编译器：浮点精度调优

- **优化及改进不同精度模式下的浮点编译算法**
- 除法转乘法，RCP指令精度优化

优化点

- **乘法指令替换除法，可以使能流水线化**
- **调节牛顿迭代，按需调整结果精度**

精度选项

(1) -freciprocal-math or -ffast-math

(2) -mrecip=x:y or -mrecip.

X指定rcp操作，如div，sqrt

Y指定牛顿迭代次数，值是0~9

场景描述（单精度）：

```
fdiv    s8, s8, s10

(~ 11cycles)
```

预期优化结果（default）：
两次牛顿迭代

```
frecpe  s2, s8
frecps  s4, s2, s8
fmul    s2, s2, s4
frecps  s4, s2, s8
fmul    s2, s2, s4
fmul    s10, s2

(pipeline with II=6)
```

预期优化结果（sdiv=1）：
降低精度，一次牛顿迭代

```
frecpe  s2, s8
frecps  s4, s2, s8
fmul    s2, s2, s4
fmul    s10, s2

(pipeline with II=4)
```

HUAWEI

毕昇编译器：浮点精度调优

- **优化及改进不同精度模式下的浮点编译算法**
- 乘加指令合并，通过控制（复数）乘加指令生成数量，进行精度优化

乘加指令合并

场景描述（a*b+c）：

选项：-ffp-contract=on
预期优化结果：

```
fmul    s1, s1, s2
fadd    s0, s0, s1
```

→

```
fmadd s0, s1, s2, s0
```

矢量化 预期优化结果：

```
fmul    s1, s1, s2
fadd    s0, s0, s1
```

→

```
fmla    v4.4s, v2.4s, v1.4s
fmla    v5.4s, v3.4s, v1.4s
```

复数乘加指令合并

场景描述 g = (a+bi)*(c+di); 其中
e = a + bi, f = c+ di

使用fcmla指令:
g = fcmla (e, f, 0),
g = fcmla (e, f, 90)

```
fmul    v5.2d, v0.2d, v1.2d
fmul    v6.2d, v0.2d, v2.2d
fmls    v5.2d, v2.2d, v18.2d
fmla    v6.2d, v1.2d, v18.2d
```

↓

```
movi    v1.2d, #0000000000000000
fcmla   v1.2d, v0.2d, v3.2d, #0
fcmla   v1.2d, v0.2d, v3.2d, #90
```

HUAWEI

目录

HUAWEI

毕昇编译器：新版本

- 毕昇编译器 1.3.3 版本将于2021.06.30正式发布，包含以下新特性

  - 支持基于Structure Peeling的特性增强及指针压缩优化
  - 完善Fortran2003/2008语言特性
  - 新增大量优化特性： 支持unroll and jam，loop distribution优化增强，软件预取增强， 静态分支优化改善，循环矢量化特性增强等
  - Autotuner 特性增强与完善，改善调优时间
  - 修复前一版本中fortran及编译器中后端的若干bugs

毕昇编译器：资源获取及社区支持

交付件类型	链接	使用说明
软件包	https://www.hikunpeng.com/developer/devkit/compiler	毕昇编译器页签点击”毕昇编译器软件包下载”，下载后解压使用
sha256	https://www.hikunpeng.com/developer/devkit/compiler	毕昇编译器页签点击”毕昇编译器 sha256”，用于对比完整性校验结果
软件许可	https://www.hikunpeng.com/developer/devkit/compiler	毕昇编译器页签点击”毕昇编译软件许可”查看软件使用许可
文档	https://www.huaweicloud.com/kunpeng/software/bishengcompiler.html	毕昇编译器用户指南
	https://support.huaweicloud.com/fg-autotuner-kunpengdevps/kunpengbisheng_20_0002.html	Autotuner特性指南
问题讨论	https://bbs.huaweicloud.com/forum/thread-70301-1-1.html	鲲鹏社区论坛，可发帖提问

HUAWEI

毕昇编译器：资源获取及社区支持

相关链接：

鲲鹏论坛编译器版块：

https://bbs.huaweicloud.com/forum/forumdisplay-fid-928-orderby-lastpost-filter-typeid-typeid-1642.html

鲲鹏开发套件：

https://www.hikunpeng.com/developer/devkit/compiler

鲲鹏首页：

https://www.hikunpeng.com/

HUAWEI

总结

- 针对不同的场景，不同的应用特点，使用不同的编译优化手段。
- 编译优化的代价与收益权衡，需要综合考虑性能收益，代码体积，编译时间，可调试性等多方面因素；浮点优化场景需要平衡应用性能及结果精确度。
- 语言生态构建的持续性，语言标准的不断演进，及新特性支持。
- 软硬件结合，通过软件及硬件的协同优化，最大化的发挥硬件算力。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home and
organization for a fully connected,
intelligent world.

HUAWEI

QCon+ 案例研习社

扫码学习大厂案例

学习前沿案例，向行业领先迈进

40 个
热门专题
—
行业专家把关内容筹备，
助你快速掌握最新技术发展趋势

200 个
实战案例
—
了解大厂前沿实战案例，
为 200 个真问题找到最优解

40 场
直播答疑
—
40 位技术大咖，每周分享最新
技术认知，互动答疑

365 天
持续学习
—
视频结合配套 PPT
畅学 365 天