

基于资源适配的软件无线电应用自动部署算法

宋春磊, 高 博

(解放军信息工程大学 信息工程学院, 河南 郑州 450001)

摘 要: 针对当前软件无线电应用自动部署算法在运行环境约束方面的考虑不足, 提出了一种更为通用的系统模型, 将应用的处理资源需求、内存资源需求、带宽资源需求和运行环境需求都纳入到应用部署过程中; 采用基于可变窗长的部署算法以达到最小化系统资源占用的目的. 仿真结果表明, 运行环境约束对应用的部署具有重要影响. 另外, 该算法窗长越大, 算法性能越好.

关键词: 软件无线电; 应用部署; 系统模型; 运行环境; 资源占用

中图分类号: TN014

文献标识码: A

文章编号: 1000-7180(2017)10-0076-05

Automatic Deployment Algorithm of Software Radio Application Based on Resource Adaptation

SONG Chun-lei, GAO Bo

(College of Information System Engineering, The PLA Information Engineering University,
Zhengzhou 450001, China)

Abstract: Aiming at the limitation of the automatic deployment algorithm of software radio application, the paper puts forward a more general system model which integrates the processing resource requirement, memory resource requirement, bandwidth resource requirement and operating environment requirement into the application Deployment process; using a variable window-based deployment algorithm to achieve the purpose of minimizing system resource consumption. The simulation results show that the constraint of runtime environment has important influence on the application deployment. In addition, the larger the window length is, the better the algorithm performance is.

Key words: software defined radio; application deployment; system model; operating environment; resource consumption

1 引言

软件无线电(Software Defined Radio, SDR)的特点是将整个信号处理流程在软件上实现^[1-2]. SDR的提出使得用户可以通过调整或切换无线接入技术来适应当今和未来无线电环境的变化, 大大增加了无线系统的灵活性.

SDR 应用是指实现特定功能的软件组件集合^[3], 也可以理解为连续地处理和传播实时数据的一组并发过程. 将特定的组件集合部署到 SDR 平台

的处理器上, 并加载运行, 就实现了一个 SDR 应用.

随着近年来对 SDR 的研究不断深入, SDR 平台的结构开始向着异构、网络化的方向发展; 同时, 应用的种类也日趋繁多, 不同环境下应用的需求也相差甚远. 面对着有限的系统资源和日益复杂的应用环境, 如何在实现应用功能的同时使系统资源损耗最小是当下迫切需要解决的问题. 为了解决这个问题, 文献[4]提出了一种芯片级专用处理器波形部署算法, 但是该算法使用范围较窄, 应用部署的粒度较大, 资源利用率较低; 文献[5]提出了一种计算资

源管理框架,该框架能够较为高效地对系统资源进行管理,但它并没有考虑处理过程中系统的内存损耗以及组件通信时的处理损耗;随后文献[6]针对计算资源管理框架存在的不足,将内存损耗纳入到了系统模型中,但文献[5-6]都没有考虑实际应用场景中不同类型的软件组件对其运行环境的真实需求。

本文提出了一种基于资源适配的软件无线电平台自动部署算法。

2 系统模型

2.1 平台模型

典型的软件无线电硬件平台是一个由多种处理器基于多种互联机制构成的分布式异构处理平台,处理器为应用软件组件提供必要的运行环境和处理能力,互联机制则为不同处理器中两个组件之间的交互提供实时通信服务^[7]。这里,处理器包括 CPU、DSP 以及 FPGA 等多种类型,互联机制则包括点到点、总线等多种类型。因此,平台建模需要考虑处理设备的处理资源和内存资源,以及互联机制的通信资源。另外,应用组件在运行的过程中对处理器类型、编程环境及软件环境都具有较强的依赖性。这里,处理器类型包括处理器种类、厂商、型号等;编程环境包括编译工具和编译语言等;软件环境包括核心框架以及通讯中间件的版本等。这也是平台模型中需要加入的必要元素。

假设软件无线电平台 P 具有 m 个处理器,那么我们可以将 P 表示为一个长度为 m 的处理器序列,即:

$$P = [P_1, P_2, \dots, P_m] \tag{1}$$

在平台模型中,我们用主频来统一衡量处理器的处理能力,单位为 GHz,用 P^C 表示;处理器的内存资源用 P^M 表示,单位为 kB;处理器之间的通信带宽资源用 P^B 表示,单位为 MBPS。则平台的处理能力可以建模为一个一维矢量,即:

$$P^C = [P_1^C, P_2^C, \dots, P_m^C] \tag{2}$$

式中, P_i^C 表示处理器 P_i 的主频大小

相应地,处理器的内存资源建模为:

$$P^M = [P_1^M, P_2^M, \dots, P_m^M] \tag{3}$$

处理器之间的通信带宽资源建模为一个二维矩阵:

$$P^B = \begin{bmatrix} \infty & P_{12}^B & \dots & P_{1m}^B \\ P_{21}^B & \infty & \dots & P_{2m}^B \\ \vdots & \vdots & \ddots & \vdots \\ P_{m1}^B & P_{m2}^B & \dots & \infty \end{bmatrix} \tag{4}$$

式中, P_{ij}^B 表示处理器 P_i 与处理器 P_j 之间的通信带宽,对角线为 ∞ 表示单个处理器内部的通信带宽为无穷大。

我们将影响应用与处理器映射关系的处理器类型、编程环境和软件环境统一定义为应用部署过程中的运行环境约束,用 P^R 表示,则运行环境约束可以建模为一个二维矩阵,用横坐标表示处理器序号,纵坐标表示三种运行环境约束,即:

$$P^R = \begin{bmatrix} P_{11}^R & P_{12}^R & \dots & P_{1m}^R \\ P_{21}^R & P_{22}^R & \dots & P_{2m}^R \\ P_{31}^R & P_{32}^R & \dots & P_{3m}^R \end{bmatrix} \tag{5}$$

式中, P_{1i}^R 表示处理器 P_i 的处理器类型约束; P_{2i}^R 表示处理器 P_i 的编程环境约束; P_{3i}^R 表示处理器 P_i 的软件环境约束。

图 1 和表 1 分别给出一个 SDR 示例平台模型及其各种属性信息。

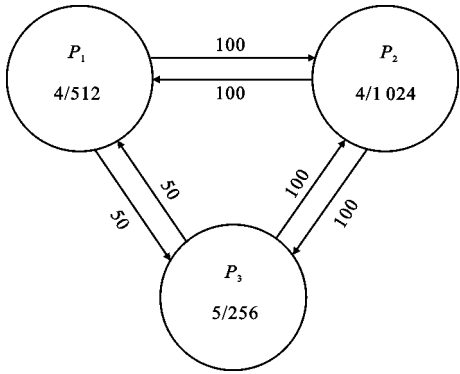


图 1 SDR 示例平台模型

表 1 SDR 示例平台属性信息

	P^C	P^M	P_1^R	P_2^R	P_3^R	P^B
P_1	4	512	CPU	C 语言	1.0	$P_{12}^B = 50$
P_2	4	1 024	FPGA	VHDL	1.0	$P_{23}^B = 100$
P_3	5	256	DSP	C++	1.0	$P_{13}^B = 100$

2.2 应用模型

一个 SDR 应用通常由运行于多个处理器的软件组件根据一定的连接关系装配组成,每一个软件组件负责特定的算法功能,任意两个组件端口之间的连接关系代表了数据传输需求。本文利用有向无环图(Directed Acyclic Graph, DAG)^[8]描述 SDR 应用,图中节点表示应用的组件,有向边表示组件间的连接关系。为了将应用合理地部署到硬件平台上,需要详细描述应用中的组件需求以及组件间的连接关系需求。

假设一个 SDR 应用 A 由 n 个组件及其连接关

系构成,那么将其建模为长度为 n 的组件序列,即:

$$A = [C_1, C_2, \dots, C_n] \tag{6}$$

与平台模型相对应,将组件的处理资源需求建模为一个一维矢量:

$$C^C = [C_1^C, C_2^C, \dots, C_n^C] \tag{7}$$

组件的内存需求建模为

$$C^M = [C_1^M, C_2^M, \dots, C_n^M] \tag{8}$$

组件间的连接关系对通信带宽的需求建模为

$$C^B = \begin{bmatrix} 0 & C_{12}^B & \dots & C_{1n}^B \\ 0 & 0 & \dots & C_{2n}^B \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{9}$$

式中, C_{ij}^B 表示组件 i 发送数据到组件 j 需要的最小通信带宽. 在有向无环图中,若节点 i 向节点 j 发送数据,则 $i < j$. 所以,式 (9) 为对角线为 0 的上三角矩阵.

与平台模型中运行环境约束对应的组件的运行环境需求建模为

$$C^R = \begin{bmatrix} C_{11}^R & C_{12}^R & \dots & C_{1n}^R \\ C_{21}^R & C_{22}^R & \dots & C_{2n}^R \\ C_{31}^R & C_{32}^R & \dots & C_{3n}^R \end{bmatrix} \tag{10}$$

式中, C_{1i}^R 表示组件 C_i 对处理器类型的需求; C_{2i}^R 表示处理器 C_i 对编程环境的需求; C_{3i}^R 表示处理器 C_i 对软件环境的需求. 图 2 和表 2 分别给出了一个 SDR 示例应用的 DAG 图表示及其相应的属性信息.

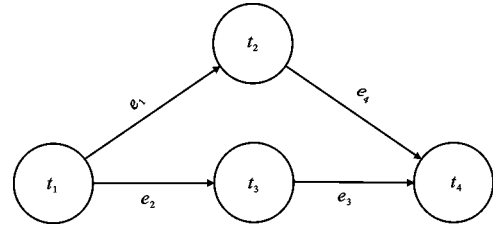


图 2 SDR 示例应用的 DAG 图

表 2 SDR 示例应用的属性信息

	C^C	C^M	C_1^R	C_2^R	C_3^R	C^B
t_1	0.2	10	CPU	C++	1.0 版本	e_1 10
t_2	0.3	30	CPU	C++	1.0 版本	e_2 15
t_3	0.5	60	DSP	C++	1.0 版本	e_3 10
t_4	0.2	15	DSP	C++	1.0 版本	e_4 20

3 基于可变窗长的部署算法

由于在 SDR 中有很多优化标准,例如满足实时性约束、优化能耗等,本文采用一种通用的部署

算法—— t_w 部署算法. 该算法能够动态地适应软件无线电环境的变化,实现软件无线电系统的资源适配.

3.1 t_w 部署算法

t_w 部署算法是一种窗口动态规划算法,这里 w 指窗长. 假设 SDR 平台中有 N 个处理器,待部署的应用中包含 M 个组件,那么在该算法中,应用的部署过程可抽象为 $N * M$ 矩阵的最优路径选择的过程,图中节点表示将组件部署在处理器 P_j 上,处理器序列 $[P_{k(0)} P_{k(1)} \dots P_{k(w)}]_i$ 表示与节点 $(P_{k(1)}, f_i)$ 关联的 w 路径, $P_{k(0)}$ 是第 $i-1$ 步时的初始处理器, $P_{k(w)}$ 是第 $i+w-1$ 步时的终点处理器. t_w 部署算法可以分解为三个部分:预处理、中间处理和后处理.

3.1.1 预处理

该算法的第一部分是部署组件 f_1 ,对于平台系统中的任意一个处理器 $P_{k(0)}$,计算将组件 f_1 部署到处理器 P_i 上的代价值 $CT(P_{k(0)}, f_1)$,并将该代价值记录在节点 $(P_{k(0)}, f_1)$ 上. $P_{k(l)}$ 表示处理器号,其中 $l \in 0, 1, 2 \dots w$.

3.1.2 中间处理

当 $w = 1$ 时,对于节点 $(P_{k(1)}, f_i)$,得到的处理器序列为 $(P_1 P_{k(1)}), (P_2 P_{k(1)}), \dots, (P_N P_{k(1)})$. 用 $WT(P_{k(0)}, P_{k(1)})_i$ 表示通信边 $(P_{k(0)}, P_{k(1)})_i$ 的权重,该权重代表了在组件 f_{i-1} 部署在处理器 $P_{k(0)}$ 的条件下,组件 f_i 部署在处理器 $P_{k(1)}$ 上所需的代价. t_1 部署算法就是计算每个通信边的累积代价值 $CT(P_{k(0)}, P_{k(1)})_i = CT(P_{k(0)}, f_{i-1}) + WT(P_{k(0)}, P_{k(1)})_i$,令

$$k(0)^* = \underset{k(0)=0,1,2,\dots,N}{\operatorname{argmin}} \{CT(P_{k(0)}, P_{k(1)})_i\} \tag{11}$$

则通信边 $(P_{k(0)}, P_{k(1)})_i$ 就是部署的最优路径,标记此条通信边并将累积代价值记录在节点 $(P_{k(1)}, f_i)$ 上.

当 $w > 1$ 时,对于节点 $(P_{k(1)}, f_i)$,算法需要分析每一条起始于第 $i-1$ 步的任意节点,途经 $(P_{k(1)}, f_i)$,终止于第 $i+w-1$ 步的任意节点的所有路径,并计算相应的累积代价. 对比 t_1 部署可以知道,当

$$\{k(0)^*, k(2)^*, \dots, k(w)^*\} = \underset{k(0)=0,1,2,\dots,N}{\operatorname{argmin}} \{CT(P_{k(0)}, P_{k(1)} \dots P_{k(w)})_i\} \tag{12}$$

时,可以得到最优路径 $[P_{k(0)^*} P_{k(1)^*} \dots P_{k(w)^*}]_i$. 若此时 $i < M-w+1$,标记 $(P_{k(0)^*}, P_{k(1)^*})_i$ 并将累积代价记录到节点 $(P_{k(1)^*}, f_i)$ 上;否则,标记整条路径

$[P_{k(0)} * P_{k(1)} * \dots * P_{k(w)}]_i$ 并将累积代价记录到节点 $(P_{k(1)}, f_{M-w+1})$ 上。

在第二部分中, t_w 部署算法可以并行处理第 i 步的全部 N 个节点, 第 i 步处理完成后, 继续进行第 $i+1$ 步的处理, 直到 $i \geq M-w+1$, 此时, 转入第三部分。

3.1.3 后续处理

算法的第三部分主要是针对第一部分和第二部分的结果进行后续处理。首先, 标记第 $M-w+1$ 步时拥有最小累积代价值的节点, 然后以此节点为起始位置, 回溯所有被标记过的通信边, 同时标记此过程中遍历的所有节点, 最终得到的结果就是在特定环境和指定窗长的条件下的最优部署方案。

从算法的第二部分与第三部分可以看出, 窗长 w 的大小决定了部署决策的局部性水平。当 $w=1$ 时, 对于每一个节点, 每一步都只有 N 种可能, 这也使得我们最后得到的部署决策只有 N 种可能。而当 $w=M-1$ 时, 算法会分析所有的 N^M 种可能, 更确切地说, t_{M-1} 部署算法对每一步的每一个节点都要计算所有通过该节点的路径的累积代价, 并在其中选择代价最小的一条。

同时, 窗长 w 的大小也决定了算法的计算效率和部署方案的性能: 窗长越大, 算法的复杂度越高, 但部署方案的性能也越好。

3.2 代价函数

本文以最小化占用资源总和为目的提出式(13)~(16)所示的代价函数。定义 $WT(P_{k(l-1)}, P_{k(l)})_h$ 为计算代价、通信代价与内存开销代价的总和, 即:

$$WT(P_{k(l-1)}, P_{k(l)})_h = WT_{\text{comp}}[P_{k(l-1)}, P_{k(l)}]_h + WT_{\text{comm}}[P_{k(l-1)}, P_{k(l)}]_h + WT_{\text{mem}}[P_{k(l-1)}, P_{k(l)}]_h \quad (13)$$

式中,

$$WT_{\text{comp}}[P_{k(l-1)}, P_{k(l)}]_h = \begin{cases} c_h / C_{k(l)}^{<k(l), h>}, c_h / C_{k(l)}^{<k(l), h>} \leq 1 \\ \infty, c_h / C_{k(l)}^{<k(l), h>} > 1 \end{cases} \quad (14)$$

$$WT_{\text{comm}}[P_{k(l-1)}, P_{k(l)}]_h = \begin{cases} \sum_{j=1}^{h-1} b_{jh} / B_{P_{k(l-1)}^{<k(l), h>}, P_{k(l)}^{<k(l), h>} \leq 1 \\ \infty, B_{P_{k(l-1)}^{<k(l), h>}, P_{k(l)}^{<k(l), h>} > 1 \end{cases} \quad (15)$$

$$WT_{\text{mem}}[P_{k(l-1)}, P_{k(l)}]_h = \begin{cases} m_h / M_{k(l)}^{<k(l), h>}, m_h / M_{k(l)}^{<k(l), h>} \leq 1 \\ \infty, m_h / M_{k(l)}^{<k(l), h>} > 1 \end{cases} \quad (16)$$

式中, $C_{k(l)}^{<k(l), h>}$ 表示在节点 $(P_{k(l)}, f_i)$, 处理器 $P_{k(l)}$

剩余的 计算 资源; $B_{P_{k(l)}^{<k(l), h>}, P_{k(l)}^{<k(l), h>}}$ 表示在节点 $(P_{k(l)}, f_i)$ 处理器 $(P_{k(l)}, f_i)$ 和处理器 $P(f_j)$ 之间剩余的 通信 带宽 资源; $M_{k(l)}^{<k(l), h>}$ 表示在节点 $(P_{k(l)}, f_i)$ 处理器 $(P_{k(l)}, f_i)$ 剩余的 内存 资源。

3.3 复杂度分析

t_w 部署算法的复杂度和取决于所选取的代价函数的复杂度, 假设计算将组件 f_i 部署到处理器 $P_{k(l)}$ 上的复杂度是不变的, 且不是 i 或者 $k(l)$ 的函数, 那么, 定义这个复杂度为代价函数的复杂度, 用 ccf 表示。节点 $(P_{k(l)}, f_i)$ 的复杂度则可以表示为

$$(N + N^2 + \dots + N^w) \times ccf = N \times \frac{N^w - 1}{N - 1} \times ccf \quad (17)$$

t_w 部署算法从预处理到得到后处理需要经过 $M-w$ 步, 且每一步都有 N 个节点, 所以 t_w 部署算法的复杂度为

$$\text{complexity}(t_w) \approx (M - w) \times N^2 \times (N^w - 1) / (N - 1) \times ccf$$

4 仿真分析

本节通过仿真实验分析 t_w 部署算法在不同平台环境中的性能以及窗长大小对算法性能的影响, 并将结果与文献[6]所采用的解决方案进行比较。算法性能可以用实际部署代价与最优部署代价的百分比值 x 来衡量, $x=100$, $0 < x < 100$, $x=0$ 分别代表该部署结果是最优, 次优和非法的。

4.1 仿真方案

本次仿真实验构建了表3所示的两种仿真场景, 其中, 场景1对应于文献[6]中采用的部署策略, 而场景2是本文针对其不足提出的改进策略。

表3 仿真场景

仿真场景	是否考虑运行环境约束
场景1	否
场景2	是

此外, 设置了如图3所示的两套 SDR 平台, 平台1中的3个处理器均为 CPU 类型, 组件编程环境只有 C 语音一种; 平台2中的3个处理器分别为 CPU、DSP 和 FPGA 类型, 组件的编程环境分为 C 语言、C++ 和 VHDL 三种。

实验中随机生成了 1 000 个表示 SDR 应用的 DAG 图, 应用的参数设置见表4。所以 DAG 图中, 共有 98.7% 占用测试平台 50% 以上的计算资源, 平均占用 75% 的资源。随机生成的 SDR 应用的总资

源需求有可能会超过测试平台所能提供的最大值，在实验时将这些 DAG 图舍弃。

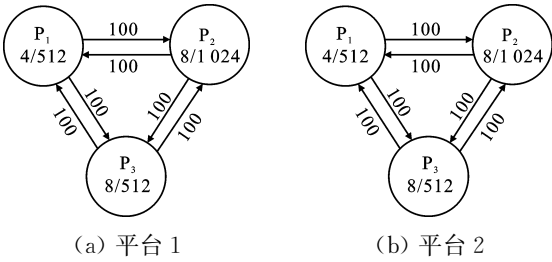


图 3 平台 1 与平台 2

表 4 SDR 应用参数设置

参数类型	参数值
组件数	12
组件处理资源需求	$U[0,1,4]$ GHz
组件带宽资源需求	$U[1,50]$ MBPS
组件内存资源需求	$U[1,256]$ KB
组件处理器类型需求	CPU/DSP/FPGA
组件编程环境需求	C/C++/VHDL

4.2 仿真结果分析

仿真实验选择的性能指标为非法部署占据的百分比，将软件组件依次部署至两套平台中，进行 1 000 次重复实验，获得性能指标的统计值，如图 4 所示。

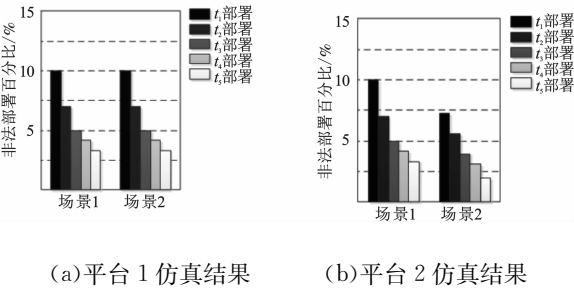


图 4 平台 1、2 的仿真结果

如图 4 所示， t_w 部署算法的性能会随着窗长 w 的增大而越来越好，这就允许我们在不同的应用需求下选择不同窗长的算法。例如，要求应用部署的非法率低于 5% 时，可以选择 t_3 算法，适当的降低算法的复杂度。

另外，对比图 4 可以看出，当我们的平台中拥有多种处理器以及多种组件编程环境时，考虑了运行环境约束的策略性能明显好于没有考虑运行环境约束的策略。因此，可以得出结论，环境约束是应用部署过程中非常重要的影响因素。

5 结束语

随着软件无线电技术的发展，应用对 SDR 平台的资源需求更加多样化，也更加具体。本文提出了一种通用的系统模型，将应用的处理资源需求，带宽资源需求，内存资源需求与运行环境需求都纳入到应用部署决策过程之中，确保应用能够分配到最适合的计算资源；另外，采用基于可变窗长的部署算法，提出最小化占用资源总和的代价函数。通过仿真验证算法性能，仿真结果表明，不同的平台环境算法的性能表现不同，并且窗长越大，算法性能越好。

参考文献：

[1] Mitola J. The software radio architecture[J]. IEEE Communications Magazine, 1995, 33(5):26-38.

[2] Buracchini E. The software radio concept[J]. IEEE Communications Magazine, 2000, 38(9):138-143.

[3] Jondral F K. Software-defined radio-basics and evolution to cognitive radio[J]. EURASIP Journal on Wireless Communications and Networking, 2005, 2005(3):275-283.

[4] 刘青, 曹凯, 徐春林. 芯片级专用处理器波形部署算法[J]. 电信技术研究, 2012, 52(2):1-7.

[5] Marojevic V, Ballesté X R, Gelonch A. A computing resource management framework for software-defined radios[J]. IEEE Transactions on Computers, 2008, 57(10):1399-1412.

[6] 吴尚峰, 唐麒, 魏急波, 等. 具有吞吐量和时延约束的软件无线电波形部署决策算法[J]. 国防科技大学学报, 2015, 60(1):136-141.

[7] 陈路辉, 吴尚峰, 吴宇, 等. 基于 SDR-4902 平台的波形部署算法设计与实现[J]. 现代电子技术, 2013, 37(21):37-41.

[8] Bajaj R, Agrawal D P. Improving scheduling of tasks in a heterogeneous environment[J]. IEEE Transactions on Parallel & Distributed Systems, 2004, 15(2):107-118.

作者简介：

宋春磊 男, (1992-), 硕士研究生. 研究方向为信息处理平台、数据库应用. E-mail: 814793089@qq. com.

高 博 男, (1972-), 博士, 副教授. 研究方向为信息处理平台、定位技术.