大型 HIS 应用系统实现中的技术选择

The Technical Choice In HIS

张哲冰 （北京交通大学计算机学院 100044）
李永昊 （北京交通大学通信学院 100044）

摘要：本文从 HIS 的应用背景出发，通过比较现有主流产品中实际采用的技术路线，提出了在大型 HIS 应用中的技术支撑应该具有的特点，并具体分析了使用 COM＋、J2EE 或. NET 作为技术选择所能够达到的预期水平。本文对于规划、开发大型 HIS 应用具有实际意义。

关键词：医院信息系统 多层应用 C/S 模式 COM＋ J2EE .Net

1 大型 HIS 应用的定义

为了适应现有应用并考虑到医院业务规模和业务范围扩展的预期要求，一个大型的 HIS 应用应该具有以下特征：

（1）医院在信息化建设上不仅含有目前面向管理的医院信息系统，同时也要含有面向临床应用的临床信息系统。同时为院际间的信息交流、资源共享做准备。

（2）12 个月的运行过程中积累 9G 以上的数据，对于以数据中心模式运营的大集中应用，应该在 12 个月的运营过程中积累不少于 40G 的数据；

（3）对于独立的运行单位，例如一个医院，在 1 个小时的工作时间内，应该有不少于 500 个用户人次的活动，并且平均有不少于 12000 笔的事务活动；

（4）系统具有分布式部署的现实需求或者可以预见的潜在需求，包括数据库部分的分布式部署、应用逻辑的分布式部署和访问点的分布式部署；

（5）系统具有基于系统的（而不是基于应用的）热备的预留能力，以及快速恢复的能力，预期完全装载 80G 的应用数据，整个系统恢复时间不应该超过 2 个小时。

具有以上特点或者满足以上要求的 HIS 应用，我们可以认为是一个大型的 HIS 应用。

2 典型应用场景

基于以上讨论，我们预期的大型 HIS 应用，将使用在以下的典型应用场景中：

一个具有 1400 张床位的医院，占地面积 45000 平方米，有 40 个以上的病区，平均日门诊量在 4000～6000 人次，平均每天总交易事务量 18 万笔，每年平均积累数据 12G 左右（不包括 PACS 系统）；整个医院已经建立以光纤为骨干的千兆网，在各个部门部署着 900 台左右的桌面计算机，可以以 100M 的速度访问中心结点；中心结点使用 2 台数据库服务器，并使用另外同样配置的 2 台服务器以热备的方式通过 SAN 进行联接，应用逻辑分布在 16 台中间服务器上，并且通过无状态的分布式部署可以随时增加、减少中间服务器数目，访问服务器和静态文件服务器分布在 32 台前端服务器上，并通过 F5 BigIP 类似的设备进行基于路由的自动负载均衡，整个结点同时提供另外 4 台独立服务器运行 EAI 应用、历史系统等；这个医院已经预期将建立另外两个分院，分别在距离该医院 15 公里和 50 公里处，租用公用链路联接，分院将同样使用中心医院的 HIS 应用系统；由于该院的学术地位，它同时还会接受来自世界各地的医学研究者的远程访问，来和该院的医生进行学术合作和交流。

以上应用场景是一个具有普遍性的虚构场景，我们下面的讨论都将以该场景为背景进行讨论。

3 可选用的技术框架讨论

3.1 两层 C/S 应用的特点分析

目前许多 HIS 应用都是基于两层 C/S 架构构建的，从系统平均响应时间来看，两层系统具有最高的响应效率，因为所有对数据的处理均在客户端本地完成，服务器仅仅提供数据存取控制，处理负荷非常均匀的分布到了广泛的客户端，因此，该结构的系统具有最高的平均响应效率。

但是，在大型 HIS 应用的场景中，采用两层结构的 C/S 应用存在以下问题：

首先，两层系统的客户端的版本控制和部署实施在大型应用中会成为一项专门的工作，它需要相当数量的系统维护人员。

其次，在传统的两层结构中，应用的业务逻辑、用户操作层甚至数据库连接都被挤压到一层应用程序中，开发人员难以将业务逻辑封装成独立的组件，综观目前存在的两层系统

都是由外观驱动业务而不是由业务驱动外观的，业务逻辑的代码并不独立于外观代码，这样就无法对稳定的业务逻辑进行重用，从而造成开发的困难。

第三、目前主流的 C/S 应用中，虽然只需要使用服务器端的数据，但是实际上还需要使用数据库服务器的联接(Connection)资源，当应用数量大规模增长后，会对后台数据服务器资源形成巨大创建/销毁连接的压力，因此一般来说两层 C/S 应用在服务器端同时承担 500 个活动客户端的时候已经非常吃力。

最后，客户端到服务器的调用是基于专门的端口和协议，具体的端口和协议与使用的数据访问驱动相关，但是这些端口和协议一般都不是为广域网环境设计的，尤其是它们一般都不是防火墙开放的端口和协议，因此，在地域分布或者在有基于广域网访问的环境中，无法适应要求。

任何事物总是具有矛盾的两个方面，以上特点也决定了两层 C/S 应用在大型 HIS 应用中的以下场景下应该采用：

(1) 对操作性能有特殊的要求，在每次操作后必须以最快的速度返回结果的场合下应该采用两层 C/S 应用；

(2) 在局部的、可控制的环境下，可以在小规模的某些应用场景下采用两层 C/S 应用；

例如挂号及门诊收费等对系统响应时间有严格要求的应用场景中，即使在 12 个小时中一直维持一个数据库联接而从不释放任何资源也是可以接受的，这种应用以采用两 c/s 应用为好。

3.2 基于 COM + 的多层应用，采用基于本地桌面程序的客户端或浏览器的客户端

3.2.1 Windows DNA 架构介绍

使用 COM + 构建多层的应用，具有以下 3 个设计理念：

(1) COM + 应用程序跨多台服务器支持多用户，也就是说，多层分布式应用的实际应用代码并不全部在客户端执行，而是在不同主机上运行的；

(2) COM + 应用程序是可扩展的，而且可以通过在机群系统中配置 COM + 应用程序来提供更好的可用性和更高的可靠性；

(3) 一个 COM + 应用程序的永久数据的状态保持在一个 COM + 支持的数据库中，如果出现错误，可以使用事务来维护这些状态。

3.2.2 基于本地桌面程序的客户端应用特点介绍

该框架采用了基于本地桌面程序的客户端，这些客户端主要用于接受用户的操作并和用户构成互动，并将用户的请求转换成 COM + 调用。由于采用了基于本地桌面程序的客户端，在部署客户端时将同样遇到部署成本、客户端成本的一些问题，尤其是 COM + 调用是防火墙敏感的，因此，基于本地桌面程序的客户端调用 COM + 应用同样是不适合复杂的、跨越网络边缘的应用场景。

由于 COM + 具有开发成本低、系统部署简单的特点，在一些需要快速完成并可能经常变更的应用上，采用 COM + 来封装应用逻辑，是非常合适的选择。

3.2.3 基于浏览器的客户端应用特点介绍

采用基于浏览器的客户端，使得对 COM + 组件的调用目前只有两种方式来完成：ASP 和 ISAPI，由于 ISAPI 是面向性能特别敏感或者有特殊要求的场合，而在大型 HIS 应用中，性能敏感的部分已经封装在 COM + 组件中，因此采用 ASP 调用 COM + 组件的方式非常适合。通过在 ASP 产生 HTML 的可视外观，并将所有涉及到数据验证、业务逻辑的部分封装到 COM + 组件内部而只通过有限的接口提供特定服务，可以将脚步代码的轻量、简单的特点和 COM + 组件高度封装、性能良好的特点有效结合，是规划大型 HIS 应用时值得考虑的架构。

3.3 基于 J2EE 或. Net 的多层应用，采用基于本地桌面程序的客户端或浏览器的客户端

3.3.1 基于浏览器的客户端应用特点介绍

这是目前被广泛采用的技术框架。由于 J2EE 和. Net 天生支持的分布式特性，尤其是都专门提供了面向 Web 表现层的技术工具——JSP + Servlet 和 ASP. net，因此在作为具体实施手段时具有更好的可操作性。

J2EE 提供了 JSP + Servlet 作为基于浏览器的表现层技术，相应的. Net 提供了 ASPX 技术。

JSP + Servlet 是作为对 EJB 的补充，JSP 最终也是在运行时被编译成为 Servlet 执行的。JSP 主要依靠标准的 HTML 标签集，也支持用户自定义的标签，因此使用 JSP 开发基于浏览器的客户端时，在表现层能力上，基本上依赖 HTML 的能力，为了完成复杂的操作界面，对程序员的要求比较高，生成效率也相对较低。

ASPX 是. NET 框架中相当重要的一部分，即是对组件技术的补充，更是. NET 和浏览器接口以及表现服务器端控件能力的技术手段。. Net 框架提供了丰富的可用用于 ASPX 的服务器端控件，例如树形、网格等等，这些控件的运行时表现在用户的浏览器中是具有相当复杂程度的 HTML 控件，但是对它们的编程都在服务器方完成。ASPX 提供一套完整的服务器端属性、方法、事件，因此，在开发具有复杂操作界面的程序时，ASPX 具有比 JSP 更好的开发效率。而且，由于 ASPX 的开发工具比 JSP 的开发工具更加高效，因此这个因素更加明显。

3.3.2 基于轻量的 Web 框架的浏览器应用

轻量的 Web 框架一般都是应用服务器厂商为弥补重型

框架的某些方面的不足——例如代码模型过于复杂等——而开发的，在和其他企业级框架整合应用时，通过仔细的权衡哪些部分放在轻量的 Web 框架中运行、哪些部分封装在 EJB 或 COM + 组件（或. Net 组件）中运行，来达到均衡利用的目的，单纯采用轻量级的 Web 框架，很难满足大型 HIS 应用的需求，尤其是在事务、安全性控制等许多方面，轻量级的框架完全没有支持。目前已经有一些产品采用了这种技术支撑框架，除了开发成本低之外，在大规模的应用中这种框架没有什么优势。

我们建议采用轻量级的 Web 框架作为门户或者访问后台业务的访问点，而且强烈建议不要将业务系统搭建在 Web 框架上。

表现层在某些情况下可以放在 Web 框架中使用纯代码的形式实现，但是在大型 HIS 应用中，可以预计将会有非常多的用户外观表现，它们之间的联接关系也会比较复杂，因此在组织大型 HIS 应用的项目时，建议表现层不仅仅依靠 Web 框架，也要尽量依靠后台平台，例如通过使用 UIML（用户无关的界面描述语言，一种 XML 语言）产生外观等。

4 我们建议的技术框架方案

4.1 前述方案对比

（1）从两层应用和 COM + 的对比来看：

由于使用 COM + 开发的多层应用其应用逻辑驻留在应用服务器上，而不是在客户机上，因此可以大大简化配置工作，只需要对少数应用服务器的部署，就可以完成整个系统使用的应用逻辑的配置。

同时，由于应用服务器一般和最终资源——例如数据库服务器——离的更近，环境更容易控制，因此可以更加高性能的访问方式，从而可以获得更好的数据库访问性能。

其次，业务逻辑将被封装成为 COM + 组件，通过提供接口而不是提供用户操作界面来提供特定的服务，使基于组件的黑盒重用成为可能。

最后，COM + 应用以事务为核心，事务提供了在错误、灾难、崩溃时，保持当前应用场景的能力，通过使用事务，可以大大提高系统的可靠性而不需要自行编写代码，尤其考虑到 COM + 的事务使用是基于部署描述的而不是基于代码的，这个特点就具有更大的价值。

. Net 作为实现了 COM 思想的更高级的框架，继承了 COM + 各个方面的优点并有了更大的发展，因此，在服务器端逻辑实现方面，. Net 框架处于更加有利的地位；但是. Net Framework 需要专门的软件支持，而 COM + 已经是 Windows2000 内置支持的服务，所以在部署方面需要根据实际情况做出决定。

（2）从. Net 和 J2EE 的对比来看：

从 J2EE 提供的服务，尤其是 EJB 容器可以提供的 CMP 和 CMR 等服务来看，J2EE 在实现大规模应用时比. Net 框架更有优势。

从开发效率看，. Net 框架提供了 VS. net，J2EE 的开发则依靠开发组织自行选择开发工具，由于 VS. net 对. Net 开发采用了完整的开发支持，开发效率比开发 J2EE 应用要高，因此在快速实施时更有优势。

从现有参照实现来看，由于 Java 阵营中已经具有许多方面的参照实现，例如 Turbine、Struts、Jetspeed、Tapestry 等等，这些参照实现的框架已经经过世界范围内各种应用的检验并且在实际应用中正在运行，具有很高的可参考性；而. Net 阵营中暂时缺乏这种成体系的参照实现供选用，主要都还是一些代码片段，大的程序框架靠开发 HIS 应用时配置的分析师确定，或在更大程度上受到 Microsoft 现有框架的限制，例如安全性控制的部分。

从系统部署规模来看，两种技术框架都可以部署为广泛分布的系统。但是关于客户状态的管理，ASP. net 相比传统框架采用了三种新的办法，基本上都是基于唯一中央结点的，而在这个方面 J2EE 具有更好的解决办法，能够适应更大规模的部署。当然，两种技术框架也都可以编写完全无状态、可以线性扩展的系统。我们认为在规划大型 HIS 应用时，HIS 应用的场景还没有达到两种框架的规模极限，因此可以淡化对部署规模的考虑。

从执行效率来看，两种框架都具有很高的效率，曾经有 Microsoft 的测试报告指出某些应用下 J2EE 的运行效率只有. Net Framework 的 50% 左右，这是在没有使用 EJB 缓存的情况的数据，不能够作为实际运行的系统的参照。EJB2. 0 相比 EJB1. 0 提供了本地接口，可以在同一个 JVM 内更高效的访问企业 Bean，而不需要频繁的借助 JNDI 和进行类型包装，因此可以通过精确调整代码和部署的组合，来获得更高性能的 J2EE 应用。

从可靠性和可用性方面来看，两种框架都支持事务，也都支持基于代码的事务和基于部署说明的事务，因此都具有足够的可靠性；从可用性方面看，两种框架都通过支持分布式部署来分担单结点风险，从而提供更高的可用性，由于. Net 框架在企业应用环境中使用时间还比较短，因此我们认为它还需要更多的成功案例来说明它的能力，而 J2EE 应用在案例支持方面非常有利。

从客户可接受程度来看，似乎客户更愿意接受 J2EE 应用，也是因为 J2EE 应用已经出现的成功案例。

因此，综合考虑，我们认为从纯技术的角度出发，J2EE 稍

微具有优势，但是从成本的角度出发. Net 更有优势。因此，将一部分稳定的、关键的业务采用 J2EE 的方式实现，同时将另一部分多变的、非关键的业务采用. Net 的方式实现，使用 Web Service 在两种封装的组件之间进行互操作，是一种切实可行的技术框架。

4.2 我们建议的技术框架

通过上述的分析我们可以看到，每种技术框架都有其优点与弱点，都有其适应于某种应用需求的地方。但同时也看出来，如果从满足我们最初提出的应用场景需求出发，并没有一个方案单独可以最好的满足应用的全方位要求。因此，我们建议的方案是，根据应用场景的要求，综合采用所有的技术。通过从各种技术方案中选择它们最合适的部分，并通过一个完整的框架将这些部分整合使用，这种方案最有希望达到大型 HIS 应用的所有方面的要求。

我们建议的技术框架如图 1 所示：

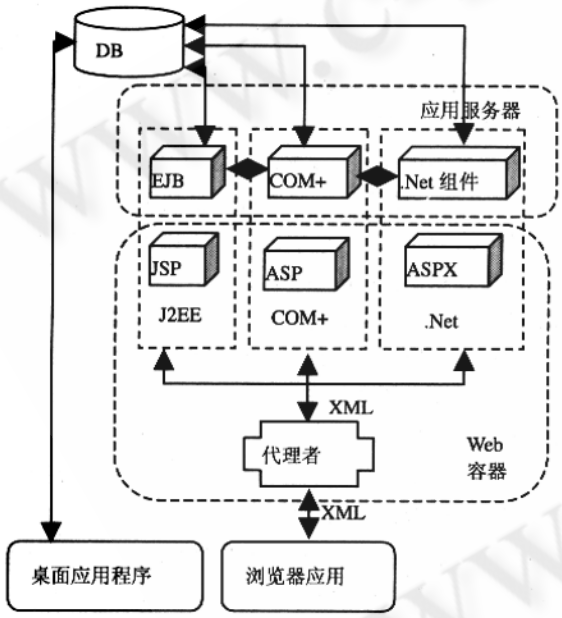

图 1　技术框架

4.3 相应团队模型

为了采用以上建议的技术框架，需要配置具有合适模型的团队。我们尤其要避免的团队模型是将产品中的某个功能模块分配给团队中的一个更小的团队后，由这个小团队从后到前全面负责的作法；我们的目的是建立一个可以长期稳定发展、团队成员分工明确的团队模型。

图 2 所示为建议的团队模型图：

业务分析师和团队管理者贯穿开发过程全局。业务分析师是精通商业需求和商业流程的业务专家，他在整个开发过程中定义产品特性和边界，并随时为开发人员解答他们对业务要求的疑问。团队管理者及协调者在整个开发过程中控制开发进度、协调团队和环境的关系、安排交流以及在某些事情上做出仲裁，团队管理者可能会同时管理着几个团队。

图 2　团队模型图

开发人员则需要按照系统技术框架的划分来水平划分。出于细化分工的考虑，可以将开发人员按照数据库程序员、应用逻辑程序员和表现层程序员进行粗略划分，根据具体采用的技术框架，可以进行更细致的划分，目的就在于每个程序员都有自己独立发展的方向，他们只是完成产品中各个功能的某一部分，通过业务分析师定义的接口的产品能够集成为完整的产品。

5 结束语

医院信息系统经历了近三十年的历程，也取得了共认的成绩，然而不可否认的是由于大环境的方方面面的制约，当前的医院信息系统存在的最大问题就是扩展性的制约，无论哪个产品与那个公司都不能快速满足不断增长的需求范围与应用深度的要求。解决这个问题需要业界全方位的努力，而技术支持是其中不可少的最重要的一部分。正确的选择技术而不是盲目的跟随当前 IT 名目繁多的新花样，这是技术支持的最基本点。本文希望能够在这方面给出一个思索，起到抛砖引玉的作用。

参考文献

1 傅征、任连仲等，医院信息系统建设与应用，人民军医出版社，2002.9。