Arquitetura para a Utilização de Computação nas Nuvens em Ambientes de Computação Pervasiva

Henrique G. G. Pereira[1], **Giovani R. Librelotto**[1]

[1]Universidade Federal de Santa Maria (UFSM)
Avenida Roraima, 1000 - Bairro Camobi - 97105-900 - Santa Maria - RS - Brasil

henriquep@acm.org, librelotto@inf.ufsm.br

Abstract. *In the past few years pervasive computing and cloud computing have appeared as very promising trends. But in order to make pervasive computing a reality, a few changes are required on the current computing environments. This article shows an architecture for the creation of pervasive computing environments using cloud computing services and ontologies for context management. The article presents this architecture and a study case implementing this architecture.*

Resumo. *Nos últimos anos tanto a computação pervasiva quanto a computação em nuvem têm surgido como tendências muito promissoras. Porém, para que a computação pervasiva se consolide, são necessárias algumas mudanças de paradigma nos ambientes atuais da computação. Esse artigo visa apresentar uma arquitetura para a criação de ambientes de computação pervasiva, utilizando os serviços disponíveis na computação em nuvem em conjunto com ontologias para gerenciamento de contexto. Ao final do artigo é exibida a arquitetura proposta e realizada um estudo de caso implementando a arquitetura proposta.*

1. Introdução

A visão mais aceita de computação ubíqua ou pervasiva é a proposta por [Weiser 1991], que mostra um mundo onde os computadores estão inseridos de forma natural no nosso cotidiano, centenas de computadores em uma sala, cada um voltado para uma tarefa específica e interagindo uns com os outros para realizar ações em nome do usuário. A computação ubíqua e pervasiva vai além das barreiras tradicionais de como, quando e onde ocorre a interação entre o homem e a máquina.

Para que essa visão se torne realidade, são necessários três componentes: computadores baratos, com baixo consumo de energia e telas convenientes, software para aplicações pervasivas e uma maneira de interligar tudo isso [Weiser 1991]. As aplicações pervasivos têm de ser proativas, isso é, descobrindo o que o usuário deseja e providenciando a ação desejada no momento correto [Loureiro et al. 2005].

A sensibilidade e o gerenciamento de contexto são dois dos desafios enfrentados pela computação pervasiva [Fournier et al. 2006]. Nos últimos anos, vários sistemas de computação ubíqua e pervasiva têm utilizado ontologias para permitir a criação de aplicações conscientes de contexto. Ontologias podem ser utilizadas para descrever vários artefatos com diferentes estruturas, indo de taxonomias simples e esquemas de metadados até teorias lógicas [Hefflin et al. 2002].

Porém, ainda existem algumas barreiras para que a computação pervasiva se torne realidade, como o alto custo de implantação e manutenção de um ambiente pervasivo, a falta de padronização desses ambientes, a grande necessidade de poder de processamento para realizar as computações necessárias em um ambiente e a falta de arquiteturas computacionais para lidar com esse problema sem comprometer a escalabilidade e flexibilidade do ambiente pervasivo [Ay 2008].

Para solucionar alguns desses problemas, o presente artigo propõe a utilização de uma arquitetura computacional para ambientes pervasivos que utilize os recursos disponibilizados pela computação em nuvem. Portanto, a computação em nuvem é apresentada na Seção 2. Posteriormente, na Seção 3, detalha-se a arquitetura proposta. Um caso de estudo é mostrado na Seção 5, com uma avaliação da metodologia proposta. A conclusão e os trabalhos futuros encontram-se na Seção 6.

2. Computação em Nuvem

A computação em nuvem, ou *Cloud Computing*, é um modelo computacional com a habilidade de permitir, de maneira ubíqua e conveniente, o acesso sob-demanda a recursos computacionais compartilhados e configuráveis. Esse modelo possui cinco características essenciais: a habilidade de escalonar recursos sob-demanda, o acesso através de uma rede, a elasticidade, alguma forma mensuração do uso dos recursos computacionais e um *pool* de recursos [Mell and Grance 2011].

Os modelos de uso e pagamento de serviços na nuvem são uma grande diferença entre o modelo de *cloud computing* e o modelo tradicional de aluguel ou compra de servidores. O usuário dos serviços na nuvem paga de acordo com a utilização dos recursos computacionais, como por exemplo horas de processamento ou volume de armazenamento utilizado [Fouquet et al. 2009]. Pode-se dividir os serviços da computação na nuvem em três modelos: Infraestrutura como Serviço ou *Infrastructure as a Service (IaaS)*, Plataforma como Serviço ou *Platform as a Service (PaaS)* e Software como Serviço ou *Software as a Service (SaaS)*.

- **Infraestrutura como Serviço**: fornecimento de processamento, armazenamento, rede e outros recursos computacionais fundamentais onde o usuário tem a capacidade de executar e implementar qualquer software [Mell and Grance 2011].
- **Plataforma como Serviço**: permite ao consumidor implementar e executar aplicações na infraestrutura da nuvem, porém o usuário não tem acesso ou controle a infraestrutura da nuvem, incluindo rede, servidores, sistema operacional ou armazenamento.
- **Software como Serviço**: todas as aplicações que são executadas na nuvem e fornecem acesso direto ao consumidor [Lenk et al. 2009]. Para [Sumter 2010] esse modelo de serviço pode ser resumido como um serviço que permite o aluguel do software ao usuário.

3. Uma arquitetura para a utilização de computação nas nuvens nos ambientes de computação pervasiva

Tradicionalmente um ambiente de computação pervasiva é composto por sensores ligados a uma rede local e que se comunicam com um concentrador. No concentrador são realizados o gerenciamento das informação de contexto, as inferências sobre o ambiente

e a transmissão desses dados para os dispositivos computacionais presentes no ambiente [Medvidovic and Malek 2007]. A Figura 1 apresenta a visão de uma arquitetura tradicional de computação pervasiva, com sensores conectados a uma rede e ligados ao computador central que processa as informações de contexto e se comunica com os dispositivos do ambiente.

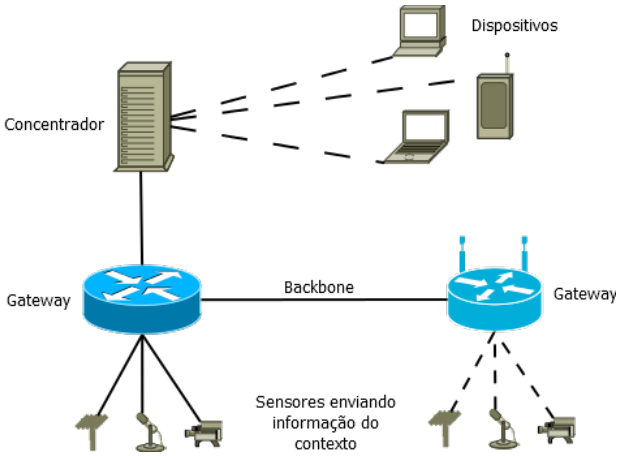

Figura 1. Arquitetura Pervasiva Tradicional

A proposta apresentada neste artigo difere das arquiteturas tradicionais por utilizar não só os recursos computacionais disponíveis no ambiente, mas recursos disponíveis através de uma nuvem computacional. A arquitetura proposta é dividida em dois módulos de componentes: um módulo pervasivo local e um módulo de serviços na nuvem computacional. O módulo local é responsável pela colecta de informação de atuação sobre o ambiente pervasivo (sensores, dispositivos, informação de contexto), enquanto o módulo remoto é responsável por realizar o processamento e as inferências utilizando a informação disponibilizada pelo módulo pervasivo local. Uma visão geral da arquitetura pode ser vista na Figura 2.

Para o desenvolvimento dessa arquitetura foram levantados os seguintes requisitos: baixo consumo de recursos, alta performance e escalabilidade, capacidade de lidar com dispositivos heterogêneos, a capacidade de lidar com aplicações heterogêneas e a utilização de ontologias para gerenciamento e representação da informação de contexto.

3.1. Módulo Local

A pilha de componentes local é composta por: sensores, dispositivos e softwares capazes de executar aplicações ubíquas, o Módulo de Monitoramento Local (MML) e o Módulo de Atuação, ou Módulo Atuador (MA), interligados por uma rede ubíqua. O Módulo local tem como objetivo reduzir o custo computacional da computação pervasiva, permitindo que o processamento pesado seja realizado na nuvem. A Figura 3 apresenta o Módulo Local.

O MML é responsável por coletar as informações disponíveis nos diferentes sensores e dispositivos presentes no ambiente pervasivo, realizar um processamento simplificado sobre as informações de contexto e emitir ordens ao Módulo de Atuação ou disponibilizar as informações de contexto para os serviços presentes na nuvem. O MA deve ser capaz de receber comandos tanto do Módulo de Monitoramento Local quanto do

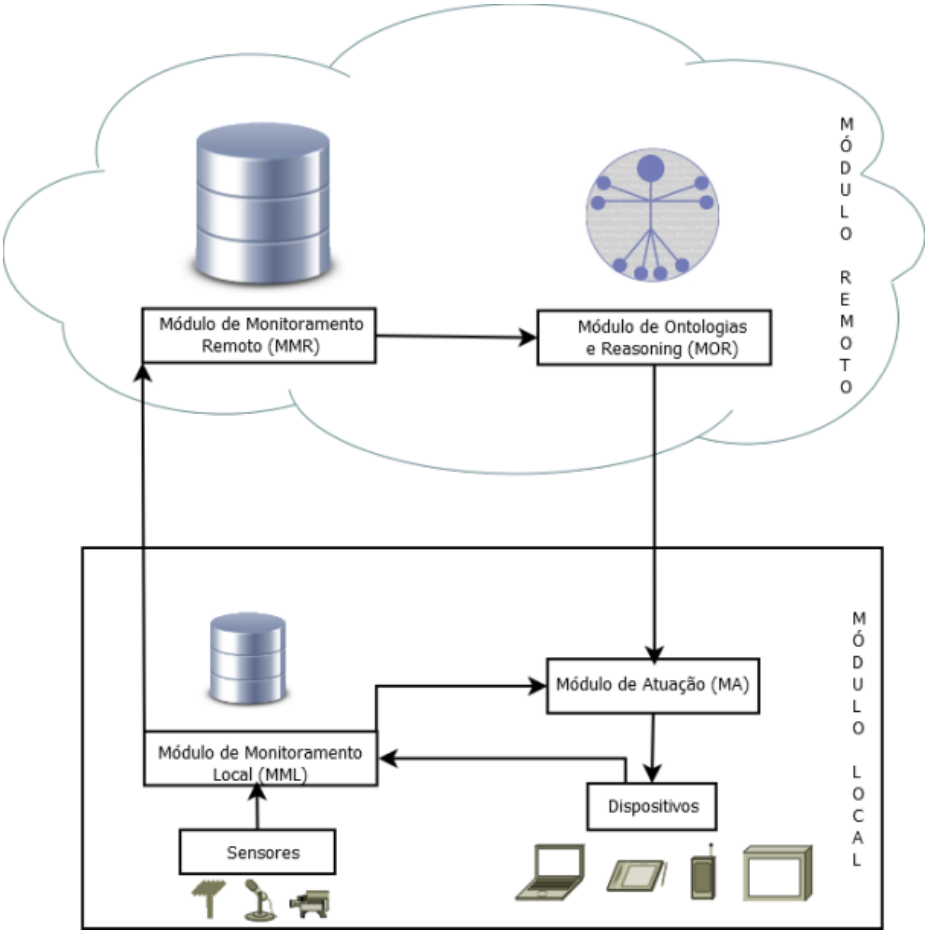

Figura 2. Visão Geral da Arquitetura Proposta

Módulo de Ontologia e Reasoning e enviar esses comandos para os dispositivos presentes no ambiente pervasivo local.

3.2. Módulo Remoto

A pilha de serviços na nuvem é composta de dois componentes: o Módulo de Monitoramento Remoto (MMR) e o Módulo de Ontologias e *Reasoning* (MOR), interligados a uma rede ubíqua. O principal objetivo do *stack* de serviços na nuvem é realizar o processamento pesado dos ambientes pervasivos locais, aumentando ou diminuíndo os recursos computacionais destinados ao MMR e ao MOR de acordo com a demanda desses ambientes. A Figura 4 apresenta o Módulo Remoto.

O MMR tem como objetivo armazenar as informações enviadas pelos MMLs das camadas pervasivas locais e enviar informações relativas a alteração de contexto para o Módulo de Ontologis e *Reasoning*. Esse módulo tem de ser capaz de armazenar e gerenciar a informação relativa ao contexto de inúmeras pilhas pervasivas locais e por isso deve suportar *multitenancy*.

O processamento das informações de contexto será realizado no MOR. Cada pilha local deverá possuir uma ontologia correspondente ao ambiente armazenada nesse módulo. Assim o MMR reportar alguma alteração de contexto, o MOR realizará o processamento dessas informações e caso seja necessário alguma ação junto ao ambiente

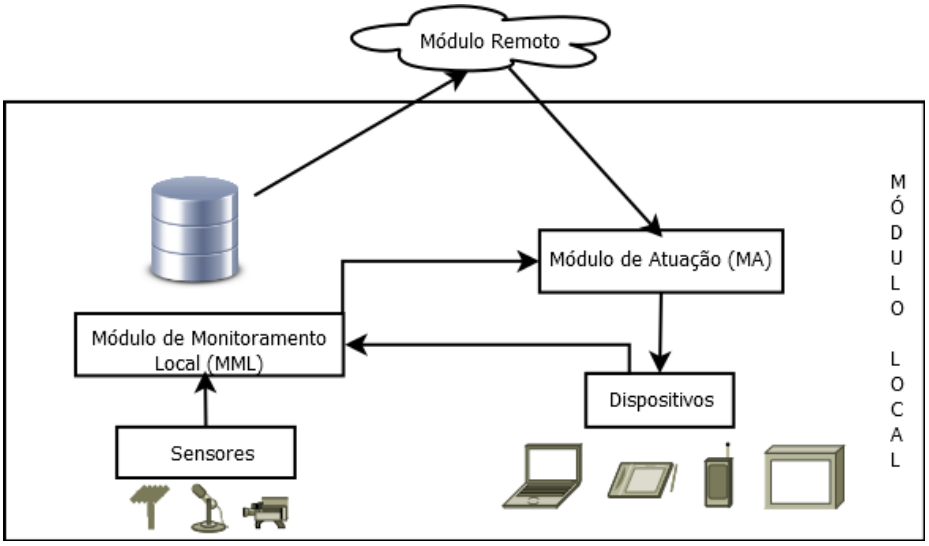

Figura 3. Visão do Módulo Local

local, enviará essa instrução diretamente para o Módulo Atuador da pilha local.

Em relação a escalabilidade, é possível que o módulo remoto, presente na nuvem computacional, escale tanto horizontalmente quanto verticalmente, sendo possível disponibilizar mais recursos para uma máquina virtual ou mais máquinas virtuais para atender a demanda dos ambientes locais.

3.3. Ontologia Para Representação de Contexto

Para realizar a representação de contexto e possibilitar o processo de inferência sobre o ambiente pervasivo, foi criada uma ontologia genérica em OWL, utilizando a ferramenta Protégé 4, que poderá ser estendida conforme a necessidade de cada sistema. A metodologia utilizada para a criação da ontologia foi a proposta por [Noy and McGuinness 2001].

O modelo da ontologia para representação de contexto é composto por quatro classes principais: *Device, Place, Software, Person*. Essas classes foram escolhidas pois são os principais componentes de um ambiente pervasivo e vão afetar diretamente o contexto do sistema.

Os relacionamentos entre as classes da ontologia foram definidos baseados na necessidade de interação do sistema pervasivo em se adaptar as mudanças de contexto. Eles foram criados para possibilitar que as inferências sobre o contexto ocorram de forma simplificada, resultando em uma maior responsividade da ontologia. Foram definidos 6 relacionamentos básicos: *connectedTo, contains, hasOperatingSystem, isAbleToRun, presentIn, runsOn*. O grafo que apresenta o relacionamento entre as classes da ontologia encontra-se na Figura 5.

4. Trabalhos Relacionados

Ao longo dos anos, várias arquiteturas e *middlewares* para sistemas pervasivos foram propostos. Optou-se por escolher trabalhos que permitissem a criação de ambientes inteligentes utilizando consciência de contexto. Foram escolhidas as arquiteturas MIDAS [Medvidovic and Malek 2007], ISAMpe [Augustin et al. 2002a], OntoHealth [Gassen 2010] e CoBrA [Chen et al. 2003].

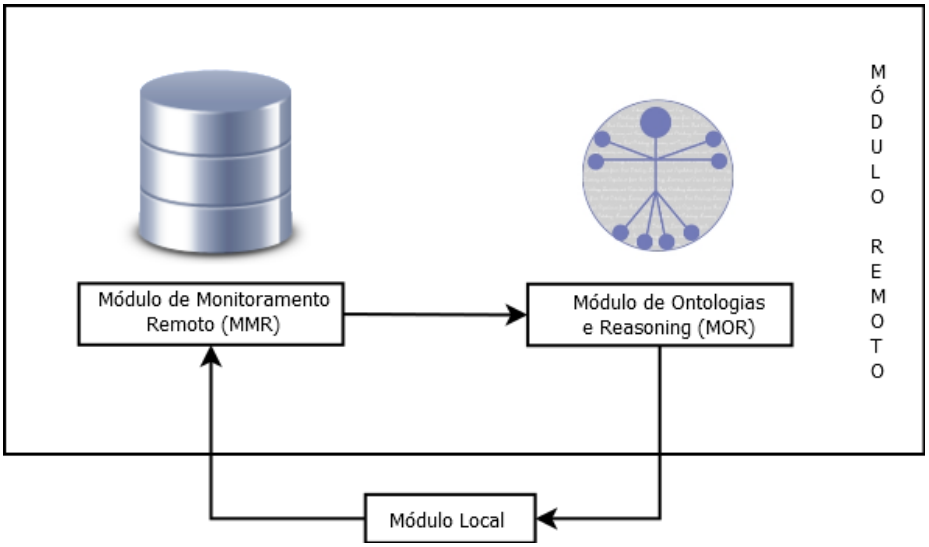

Figura 4. Visão do Módulo Remoto

4.1. CoBrA

O *Context Broker Architecture*, ou CoBrA é uma arquitetura baseada em agentes para auxiliar na computação consciente de contexto em ambientes inteligentes [Ye et al. 2007]. Ela explora a utilização de linguagens da Web Semântica para definir uma ontologia de contexto e compartilhar informações sobre contexto e inferências em cima dessa informação [Chen et al. 2003].

A informação de contexto, nessa arquitetura, é representada através de um conjunto de ontologias chamado de CoBrA-ONT, implementado utilizando OWL. Nessas ontologias ficam definidos conceitos e relações descrevendo locais físicos, pessoas, agentes de software, dispositivos móveis e eventos.

Para permitir o gerenciamento da informação de contexto e permitir o compartilhamento dessas informações, a arquitetura utiliza um *context broker*. Esse agente de contexto pode inferir informações sobre o contexto que não poderiam ser detectadas pelos sensores físicos e também é capaz de resolver inconsistências que podem decorrer de informações imprecisas vindas dos sensores.

4.2. MIDAS

A arquitetura MIDAS é uma arquitetura simples para a computação pervasiva com base na utilização de sensores [Medvidovic and Malek 2007]. Os ambientes computacionais são estáticos e os dispositivos computacionais são em sua maioria PDAs conectados a um computador central que realiza todas as computações sobre o ambiente pervasivo.

Ele é desenvolvida sobre uma família de sensores chamados MIDAS. Esses sensores se comunicam através de uma conexão sem fio com uma série de *gateways* conectados a um *hub* central que é o núcleo do ambiente pervasivo. Esse concentrador central é responsável pelo compartilhamento de informações de contexto com as aplicações pervasivas e os dispositivos computacionais presentes no ambiente.

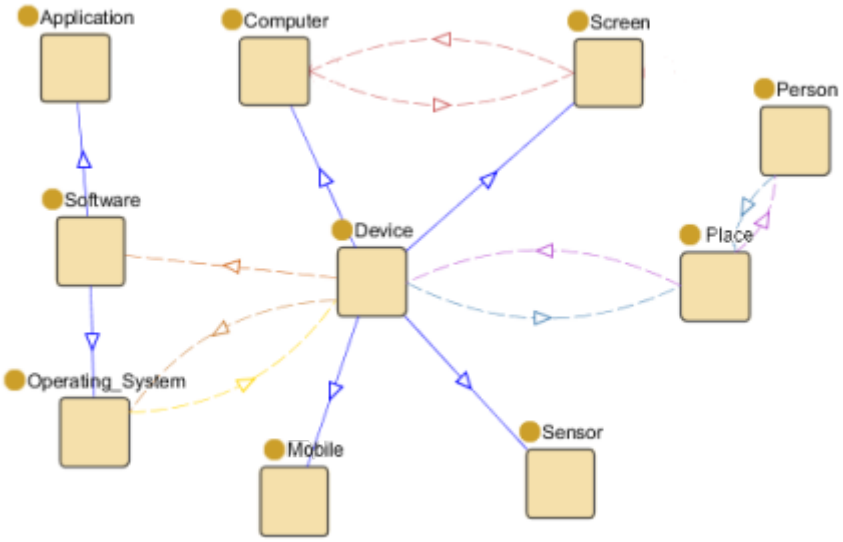

Figura 5. Grafo da ontologia de contexto

4.3. ISAMpe

O ambiente pervasivo do projeto ISAM (Infraestrutura de Suporte às Aplicações Móveis) permite que aplicações em um ambiente pervasivo sejam móveis, flexíveis e adaptáveis [Augustin et al. 2002a]. O ambiente ISAM contempla mais do que apenas um ambiente pervasivo, é um arcabouço de desenvolvimento de software, com metodologias e práticas próprias [Augustin et al. 2002b].

O ambiente pervasivo é formado por células (EXEHDAcells), que são compostas por nós fixos (EXEHDAnodes) conectados a uma rede cabeada e nós móveis (EXEHDAmob-nodes) conectados a rede sem fio. Toda célula pervasiva possui um computador central (EXEHDAbase) com a função de identificação, autenticação e liberação das funcionalidades do sistema. As várias EXEHDAcells trocam informações de contexto entre si.

4.4. OntoHealth

A arquitetura proposta por [Gassen 2010] contempla ambientes hospitalares de computação pervasiva. Ela foi desenvolvida tendo em vista que o ambiente hospitalar é diferente de uma universidade ou de uma casa, pois possui entidades ”limitadas”[Gassen 2010].

As informações de um paciente armazenadas em um Prontuário Eletrônico do Paciente (PEP), por exemplo, só podem ser compartilhadas com entidades específicas (médicos, enfermeiros e pessoas autorizadas pelo paciente a visualizar essas informações). A informação de contexto nesse sistema é gerenciada com o uso de ontologias, onde cada ontologia descreve um ambiente específico do hospital, reduzindo assim o número de entidades descritas em cada ambiente e aumentando o desempenho do processamento e a realização de inferências sobre o ambiente.

Na arquitetura OntoHealth, essas ontologias são descritas utilizando OWL. As

ontologias também fazem o uso de regras SWRL e as consultas são feitas através de SQWRL ou SPARQL.

5. Estudo de Caso

O estudo de caso buscou contemplar os aspectos diferenciados da arquitetura proposta e resultou em uma implementação funcional de um ambiente pervasivo. Optou-se por utilizar como cenário um ambiente de computação pervasiva residencial. A escolha desse cenário busca apresentar uma utilização prática da arquitetura proposta.

O ambiente de testes (Módulo Local) foi composto por quatro computadores, um dispositivo móvel, quatro sensores de ruído, quatro sensores de luminosidade, quatro sensores de presença, software pervasivo sendo executado nos computadores e nos dispositivos móveis. Todos os componentes estão conectados através de uma rede de alta velocidade. Em todos os cômodos foram instalados um computador, um sensor de ruído, um sensor de luminosidade e um sensor de presença. Cada um dos equipamentos possui um identificador único, conforme modelado na ontologia de representação de contexto.

A aplicação pervasiva que gerencia esse ambiente é capaz de adaptar o conteúdo de acordo com a capacidade técnica do dispositivo computacional que está sendo utilizado. Sendo capaz de redimensionar o tamanho da fonte, aumentar ou diminuir o volume e controlar o nível de luminosidade dos *displays*. Em um dos computadores foi instalado um software capaz de retransmitir as informações de contexto para o Módulo Remoto.

O Módulo Remoto foi implementado utilizando o serviço de computação elástica da Amazon, o Amazon EC2. Foram utilizados dois nodos, um configurado com um banco de dados para armazenar as informações de contexto e outro dotado de um *reasoner* e responsável pela realização de inferências sobre o ambiente local.

O ambiente local foi implementado utilizando sensores de software. Os componentes locais são os dispositivos, sensores, módulos de atuação e módulos de monitoramento que realizam funções a nível do ambiente pervasivo. Todos os dispositivos possuiam um navegador capaz de realizar a exibição de conteúdo criado utilizando HTML5 e foram configurados para executar automaticamente o navegador apontado para um endereço específico assim que o computador fosse iniciado. Os dispositivos também encontravam-se conectados a uma rede de alta velocidade, os computadores através de conexão Ethernet e o dispositivo móvel utilizando uma rede sem-fio.

Foram utilizados sensores simulados para representar sensores de ruído, sensores de luminosidade e sensores de presença. Cada um desses sensores foi pensado de maneira a representar fielmente um sensor real. Todos os sensores foram configurados para enviar informações diretamente ao Módulo de Monitoramento, através da rede Ethernet onde eles se encontravam conectados.

O Módulo de Monitoramento Local foi desenvolvido utilizando o framework de rede Twisted em conjunto com um banco de dados relacional SQLite. O Twisted foi responsável pela implementação de sockets capazes de realizar a conexão com o *script* utilizado para simular os sensores e pelo envio da informação do ambiente local para a pilha remota.

O Módulo de Atuação foi desenvolvido utilizando node.js e estava presente na aplicação pervasiva sendo executada pelos navegadores do dispositivo móvel. Ele foi

implementado utilizando um servidor assíncrono para responder aos comandos enviados pelo Módulo Remoto de Ontologias e Reasoning. O Módulo de Atuação recebia os comandos enviados utilizando Javascript Serialized Object Notation (JSON) e executava esses comandos nos dispositivos do ambiente local. A Figura 6 apresenta uma visão dos componentes locais utilizados no estudo de caso.

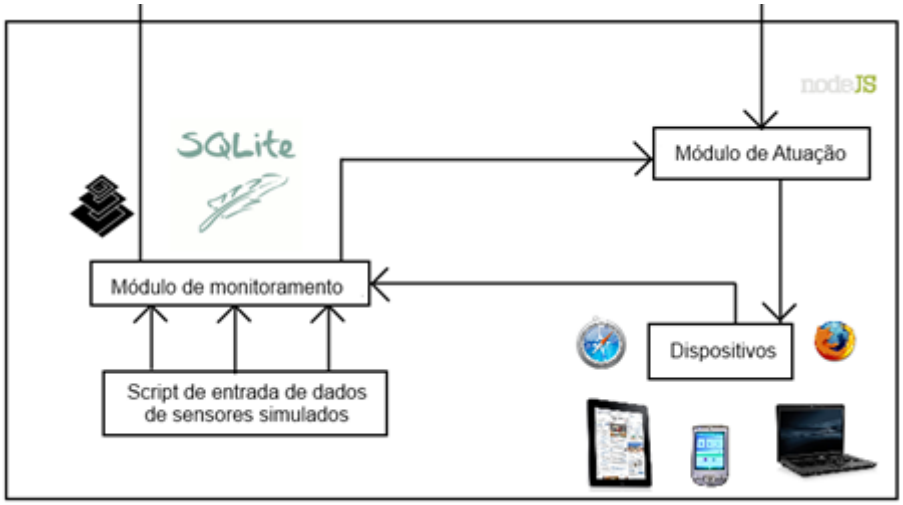

Figura 6. Visão dos Componentes Locais

5.1. Componentes Remotos

Os dois componentes remotos utilizados foram o Módulo de Monitoramento Remoto e o Módulo de Ontologias e Reasoning. Ambos os componentes foram implementados utilizando nodos presentes comercialmente no serviço de infraestrutura como serviço do Amazon EC2. Os dois módulos foram implementados utilizando soluções *off the shelf* e padrões abertos para exemplificar a capacidade do arquitetura de se adequar as soluções já existentes no mercado. A Figura 7 apresenta uma visão dos componentes remotos utilizados no estudo de caso.

A partir do modelo de ontologia fornecido pela arquitetura foi criada uma nova ontologia estendendo as características específicas dos sensores presentes no estudo de caso. Essa nova ontologia reaproveitou as classes já existentes e introduziu novas classes e relacionamentos.

O Módulo de Monitoramento Remoto consistiu em uma máquina virtual executando Oracle Enterprise Linux Release 5 juntamente com um banco de dados Oracle Database 11g. O banco de dados estava configurado para aceitar conexões provenientes do ambiente local e do Módulo de Ontologias e Reasoning e armazenar os dados em uma tabela relacional.

O Módulo de Ontologias e *Reasoning* foi implementado utilizando o software Virtuoso, desenvolvido pela OpenLink. O Virtuoso permite a execução de consultas SPARQL sobre ontologias em RDF/OWL. O Módulo de Ontologias e *Reasoning* era responsável pelo armazenamento do modelo ontológico representativo do contexto do ambiente local e pela realização das inferências em cima dessa ontologia. Como a informação foi armazenada utilizando bancos de dados relacionais tanto no Módulo de Monitora-

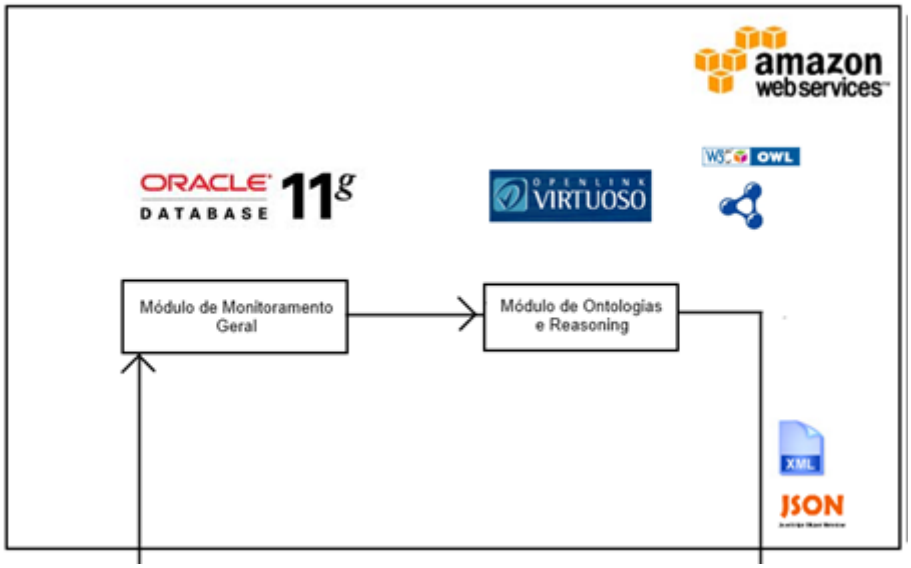

Figura 7. Visão dos Componentes Remotos

mento Local quanto no Módulo de Monitoramento Remoto, foi necessária a criação de um mapeamento para popular a ontologia.

5.2. Resultados de Desempenho

Os testes de desempenho foram realizados no Módulo de Ontologias e Reasoning conforme o Módulo de Monitoramento Remoto capturava as informações vindas do Módulo de Monitoramento Local. Foram executados 10 testes nas instâncias Pequena, Média e Extragrande do Amazon EC2. Cada teste consistiu na instanciação progressiva de entidades no modelo ontológico, variando entre 0 (zero) e 5,000,000 (cinco milhões) de instâncias, através da adição dessas informações no Módulo de Monitoramento Remoto. Ao longo da execução do teste foi executada uma mesma consulta SPARQL, onde buscou-se calcular a quantidade de vezes que essa consulta era executada.

A Figura 8 apresenta o comparativo em relação ao número médio de consultas SPARQL executadas por segundo de acordo com o aumento na quantidade de instâncias presentes na ontologia. É possível identificar que todas as instâncias apresentam uma queda no número de consultas conforme a quantidade de entidades na ontologia aumenta e, com isso, o número de consultas executadas por segundo tende a diminuir.

Durante o período de realização dos testes, os valores médios obtidos para o *round-trip time* entre Módulo Local e o Módulo Remoto ficaram em torno de 179ms, o que não foi uma surpresa, considerando a distância geográfica entre os dois módulos. A utilização de uma nuvem computacional presente na mesma rede local ou geograficamente mais próxima, pode diminuir o tempo de comunicação entre os dois módulos.

O estudo de caso possibilitou observar que com um número pequeno de instâncias na ontologia, as operações de inferência são realizadas de forma mais rápida, mas a medida que o número dessas instâncias aumenta, o poder computacional necessário para realizar as consultas aumenta e com isso o desempenho apresenta uma queda. Mesmo com 5 milhões de instâncias, a máquina Extragrande do EC2 conseguiu um elevado número de consultas por segundo, muito além do que é previsível em uma situação real.

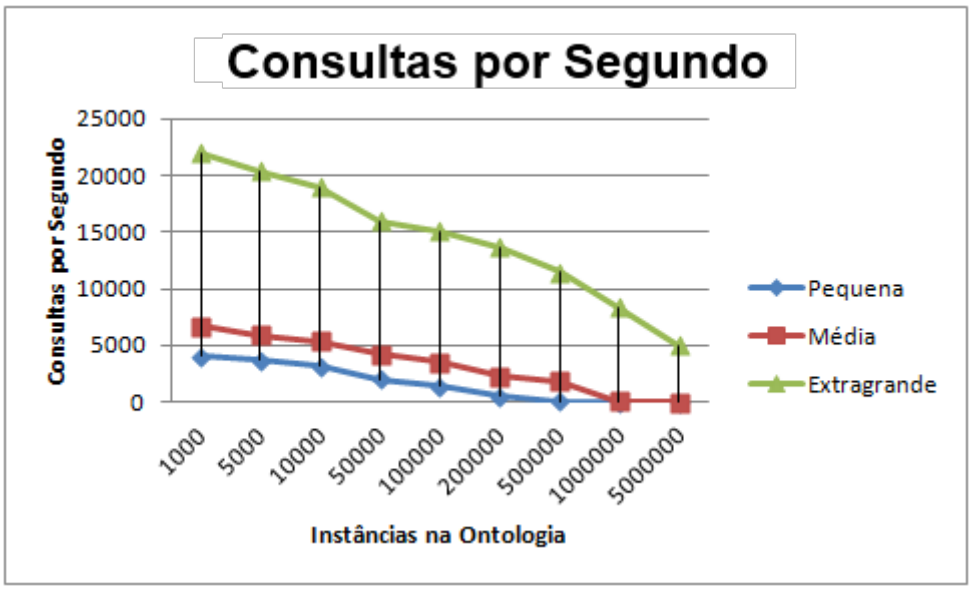

Figura 8. Número Médio de Consultas Executadas por Segundo

A utilização de tecnologias de produção, como o Virtuoso e o Oracle 11G, permitiram uma implantação rápida do ambiente de testes.

6. Conclusão

Nos dias de hoje, computadores já fazem parte do nosso dia-a-dia: possuímos telefones pequenos e rápidos, *tablets* com poder de processamento maior do que *mainframes* dos anos 1990, notebooks e computadores pessoais já não são mais artigos de luxo e podem ser encontrados sem dificuldade tanto em casa quanto no trabalho. Essa popularização dos computadores e dos dispositivos móveis faz com a visão de computação pervasiva como a proposta por [Weiser 1991] venha se tornando uma realidade.

Este artigo foi motivado pela falta de arquiteturas para o desenvolvimento de ambientes de computação pervasiva utilizando recursos disponíveis em uma nuvem computacional. A nuvem computacional poderia resolver problemas como o aumento crescente no número de dispositivos presentes em um ambiente pervasivo sem que fosse necessário interromper o ambiente pervasivo para realizar alterações de *hardware*, permitindo também uma redução dos custos associados a criação desses ambientes.

A arquitetura proposta se baseia na utilização dos recursos da *cloud computing* para atacar um problema existente e conhecido. Em relação aos módulos da arquitetura, tanto o MMR quanto o MOR podem ser observados sobre duas óticas: eles podem ser implementados utilizando os preceitos de Infraestrutura como Serviço ou podem ser disponibilizados aos usuários através de um modelo de Software como Serviço.

Referências

Augustin, I., Yamin, A. C., Barbosa, J. L. V., and Geyer, C. F. R. (2002a). Isam, a software architecture for adaptive and distributed mobile applications. In *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, ISCC '02, pages 333–, Washington, DC, USA. IEEE Computer Society.

Augustin, I., Yamin, A. C., Barbosa, J. L. V., and Geyer, C. F. R. (2002b). Isam, a software architecture for adaptive and distributed mobile applications. In *ISCC*, pages 333–338. IEEE Computer Society.

Ay, F. (2008). Context Modeling and Reasoning using Ontologies.

Chen, H., Finin, T., and Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.*, 18:197–207.

Fouquet, M., Niedermayer, H., and Carle, G. (2009). Cloud computing for the masses. In *Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities*, U-NET '09, pages 31–36, New York, NY, USA. ACM.

Fournier, D., Mokhtar, S. B., Georgantas, N., and Issarny, V. (2006). Towards ad hoc contextual services for pervasive computing. In *Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, MW4SOC '06, pages 36–41, New York, NY, USA. ACM.

Gassen, J. B. (2010). Uma metodologia para o uso de ontologias aplicadas à descrição de contexto em ambientes hospitalares pervasivos. Master's thesis, Curso de Mestrado em Nanociências - Centro Universitário Franciscano, Santa Maria, RS, Brazil.

Hefflin, J., Volz, R., and Dale, J. (2002). Requirements for a Web Ontology language. Technical report.

Lenk, A., Klems, M., Nimis, J., Tai, S., and Sandholm, T. (2009). What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 23–31, Washington, DC, USA. IEEE Computer Society.

Loureiro, E., Oliveira, L., and Almeida, H. (2005). Improving flexibility on host discovery for pervasive computing middlewares. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, MPAC '05, pages 1–8, New York, NY, USA. ACM.

Medvidovic, N. and Malek, S. (2007). Software deployment architecture and quality-of-service in pervasive environments. In *International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting*, ESSPE '07, pages 47–51, New York, NY, USA. ACM.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. *National Institute of Standards and Technology*, page 7.

Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

Sumter, L. (2010). Cloud computing: security risk. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 112:1–112:4, New York, NY, USA. ACM.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*.

Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007). Ontology-based models in pervasive computing systems. *Knowl. Eng. Rev.*, 22:315–347.