COMPUTAÇÃO DISTRIBUÍDA

Raimundo José de Araújo Macedo

COMPUTAÇÃO DISTRIBUÍDA

Raimundo José de Araújo Macedo

Resumo

A Computação Distribuída vem se desenvolvendo na academia e na indústria de informática há mais de 30 anos. No entanto, foi com o surgimento da Internet e da *World Wide Web* (WWW) que sua utilização tornou-se viável numa escala global, criando janelas de oportunidades para o desenvolvimento de novas tecnologias e serviços relacionados. Por tratar-se de uma tecnologia disponível também através da Internet, o desenvolvimento dessas tecnologias traz ao mundo em desenvolvimento, ao mesmo tempo, um desafio e uma oportunidade. Desafio na medida em que a ausência de conhecimento sobre essas novas tecnologias nos coloca cada vez mais distantes do mundo desenvolvido, requerendo, portanto, investimentos em Educação, Ciência e Tecnologia.

A oportunidade surge pelo grande espaço para novos investimentos e negócios e uma participação efetiva nesse promissor mercado global tecnológico e de serviços. Dentro desse contexto, este artigo descreve o desenvolvimento da Computação Distribuída e suas motivações iniciais, destacando as tecnologias vigentes e novos desafios tecnológicos na área. O artigo também sugere ações de governo que podem ser perseguidas para enfrentar os desafios e tirar proveito das oportunidades que se apresentam.

Abstract

The field of Distributed Computing is being developed both in academy and information technology industry for over 30 years. However, it was due to the advent of Internet and the World Wide Web (WWW) that its use became possible in a global scale, creating opportunity windows for the development of new technologies and related services. Because such a technology is also available through the Internet, its development brings to the developing world, at the same time, a new challenge and an opportunity. A challenge because the absence of knowledge about theses technologies leave us even more behind the developed countries, requiring, as a consequence, new investments in education, science, and technology. On the other hand, new opportunities arise for new businesses and investments and an effective participation in the promising global market for distributed computing technologies and related services. Following this context, this paper presents the distributed computing development process since its first motivations, highlighting the current technologies and the technological challenges ahead. This paper also suggests government actions that can be persuaded to face these new technological challenges, taking advantages of the related opportunities.

1. SITUAÇÃO ATUAL: O DESENVOLVIMENTO DA COMPUTAÇÃO DISTRIBUÍDA

No que se segue, faz-se uma retrospectiva histórica sobre o desenvolvimento da computação distribuída no mundo, desde sua motivação inicial até às tecnologias disponíveis na atualidade. Tal desenvolvimento tem se dado principalmente nos países mais desenvolvidos, a exemplo dos Estados Unidos e países da Europa. Todavia, com a globalização do acesso a Internet[1] e conseqüente disseminação global do conhecimento, torna-se cada vez mais evidente a possibilidade do engajamento dos países em desenvolvimento, a exemplo do Brasil, nesse processo.

1 Segundo relatório divulgado em 24 de novembro de 2005 pelo Ibope/NetRatings, 32,1 milhões de brasileiros - 17,3% da população acessam anualmente a Internet em casa, no trabalho, em cibercafés ou telecentros.

1.1 Uma breve retrospectiva histórica

No caminho percorrido, desde a criação dos computadores eletrônicos pelo pesquisador Von Neuman nos anos quarenta, até os dias atuais das redes de longa distância de alta velocidade (*gigabits networks*), que o processo de construção de sistemas de *software* vem sofrendo constante evolução em sua complexidade, motivado, de um lado, pela adequação à rápida evolução dos meios computacionais e de outro lado, pela necessidade de atender à crescente expansão da utilização dos computadores no mundo moderno.

Uma das principais motivações para a criação dos primeiros *mainframes*, no início dos anos sessenta, foi a possibilidade de utilização de recursos centralizados por um número de usuários através de terminais locais ou remotos. O surgimento das redes de computadores nos anos setenta e a rápida disseminação dos computadores pessoais no início dos anos oitenta possibilitaram um novo paradigma - o da computação distribuída - onde os recursos compartilhados passaram a ser também distribuídos entre um número de computadores ou dispositivos periféricos geograficamente distribuídos. Com a distribuição do poder computacional entre componentes remotos, foi possível que programas em máquinas locais (clientes) tivessem acesso a serviços remotos (servidores). Esse benefício trouxe novos desafios, tanto no aspecto distribuído de desenvolver e gerenciar *softwares*, quanto nas dificuldades inerentes ao novo ambiente: um conjunto de sistemas autônomos sem acesso a estados globais (por exemplo, uma memória RAM comum ou processador compartilhado). Dificuldades como a impossibilidade de resolver certos problemas básicos, como concordar em ações comuns na presença de falhas [4], ou mesmo a falta de ferramentas (de desenvolvimento e gerenciamento distribuído) passaram a ser uma barreira para o novo paradigma da computação distribuída.

No lado científico, as dificuldades acima relatadas motivaram uma série de pesquisas que resultaram em soluções para problemas fundamentais de uma computação distribuída, nascendo daí a especificação de mecanismos como Comunicação em Grupo, Sincronia Virtual, Consenso Distribuído, Detectores de Falhas, Difusão Confiável, entre outros [1,2,3,5,7]. Do lado da engenharia de *software* distribuído, novos modelos de desenvolvimento de software e sistemas experimentais começaram a surgir, a exemplo dos sistemas transacionais sobre banco de dados distribuídos, sistemas de *workflow*, *Computer Supported Coperative Work* (CSCW) e plataformas distribuídas orientadas a objetos. Esses novos ambientes de desenvolvimento possibilitaram uma maior utilização dos ambientes distribuídos que as novas redes e estações de trabalho cada vez mais potentes permitiam.

Entretanto, foi a partir do fenômeno da Internet, ou melhor, da *World Wide Web* (WWW), no início dos anos noventa, que a computação distribuída passou a ter relevância definitiva, a ponto de a Internet tornar-se a plataforma almejada pela maioria dos desenvolvedores de software. Em outras palavras, os projetos de *Software* passaram a incluir requisitos tais como acesso via WWW a sistemas corporativos legados, desenvolvimento de aplicações tipo cliente/servidor sobre o Transmission Control Protocol / Internet Protocol (TCP/IP) que é o protocolo padrão da Internet etc.

Adicionalmente, a proliferação da Computação Distribuída (ou dos Sistemas Distribuídos) sobre a Internet trouxe consigo a necessidade crescente de comunicação entre ambientes computacionais heterogêneos. Por exemplo, como num sistema envolvendo um determinado fabricante, seus fornecedores, distribuidores e clientes, podendo formar uma rede distribuída de grandes dimensões, envolvendo cidades e, até mesmo, países. Todavia, a comunicação entre sistemas heterogêneos apresenta diversas dificuldades resultantes do fato de que estes não foram originalmente projetados para trabalhar em conjunto, a exemplo da heterogeneidade de sistemas operacionais, protocolos, equipamentos e banco de dados.

Diante desse cenário, há uma série de requisitos do ponto de vista de engenharia de *software*, sem os quais não seria possível uma integração plena dos *softwares* desenvolvidos de forma independente e em contextos variados (geográficos e computacionais). Conseqüentemente, na Internet, faz-se necessária uma representação uniforme e abstrata de componentes[2] de *software* que permita a um usuário interagir com tais componentes sem conhecer detalhes de sua implementação e localização. Além disso, os componentes devem ser desenvolvidos de forma a possibilitar que usuários exploram dinamicamente suas estruturas, comportamentos e estados (no caso de componentes já em execução). Essas características são fundamentais para que componentes possam ser modificados depois de disponibilizados na rede global e adaptados para ambientes distintos, criando condições para uma maior produtividade na construção de *software* distribuído.

Há também outros serviços básicos necessários ao desenvolvimento de *software* distribuído, que devem ser oferecidos, preferencialmente de forma transparente, como segurança (autenticação e controle de acesso) e persistência (a capacidade de um objeto continuar a existir após a finalização da aplicação que o criou ou simplesmente o ativou).

Todas essas facilidades e serviços deveriam, idealmente, estar disponíveis através de linguagens de alto nível com mecanismos de coordenação de dados e fluxo de controle, inspeção, adaptação e disponibilização de componentes, e um ambiente de programação e integração com *Application Programming Interfaces* (APIs) tanto para implementadores de componentes como para desenvolvedores de aplicações (de preferência interfaces gráficas com visualização de estruturas de componentes, com ferramentas para integração de componentes e composição de aplicações).

Várias plataformas comerciais e experimentais desenvolvidas, principalmente nos anos noventa, resolvem boa parte desses problemas: *Common Object Request Broker Architeture* (CORBA) [9], *Distributed Component Object Model* (DCOM) [6], *Remote Method Invalation* (JAVA RMI) [8], DARWIN[19], AMOEBA [18], etc.

No entanto, nenhuma dessas infra-estruturas de desenvolvimento e gerenciamento de software distribuído visava primordialmente o ambiente Internet, que é reconhecidamente o grande desafio dos dias atuais. Para atender a essa demanda, nos anos dois mil surgiram os chamados

2 Um componente pode representar uma entidade básica como um processo (programa seqüencial em execução) ou um objeto (que possui dados privados e métodos de acesso), ou mesmo um conjunto composto dessas entidades básicas que no todo forneçam determinado serviço a rede.

Serviços *Web* (*Web Services*) e as arquiteturas de *software* orientadas a serviço (no caso, serviços *Web*) [14,20]. Por utilizarem a infra-estrutura já disponível para a WWW (como o protocolo *Hyper Text Transfer Protocol* - HTTP), amplamente usada para armazenamento e recuperação de páginas *Web*, e também por incorporarem os conceitos de *middlewares* distribuídos, como o CORBA, os serviços *Web* nasceram com o intuito de possibilitar uma integração e disponibilização global das computações distribuídas. Entretanto, sendo uma infra-estrutura global para a implementação de sistemas distribuídos, os Serviços Web por si só não resolvem todos os desafios tecnológicos, necessitando de outras tecnologias complementares para a exploração plena da Computação Distribuída, como, a utilização de computação paralela, o compartilhamento de conteúdos e a colaboração de forma generalizada (denominaremos essas novas tecnologias de tecnologias integradoras). Nesse sentido, nos últimos anos têm aparecido algumas tecnologias integradoras que possibilitam a organização de recursos computacionais (computadores) dispersos na Internet de modo facilitar seu uso de forma transparente por usuários ou *softwares* na Rede. Dentre essas tecnologias, destacam-se os sistemas *peer-to-peer*, as grades computacionais (*grid computing*), os aglomerados (*clusters*) e os agentes móveis. Diante da enorme tendência de utilização dos serviços *Web* (como veremos adiante), essas tecnologias integradoras também estão cada vez mais sendo desenvolvidas com base na infra-estrutura disponibilizada pelos serviços *Web*. No que se segue, inicialmente descrevemos a infra-estrutura de serviços *Web* e, em seguida, abordaremos brevemente essas tecnologias integradoras, seus potencias e desafios.

1.2 As Tecnologias do Momento

1.2.1 Serviços Web

Serviços *Web* são utilizados para a construção de aplicações distribuídas na Internet e representam uma evolução das tecnologias *Web*, que em seu início foram criadas apenas para construir *Web sites* - onde textos ou páginas codificados em *HyperText Markup Language* (HTML) são armazenados em servidores e lidos por usuários através de programas chamados *Browsers*. Numa infra-estrutura ou arquitetura orientada a serviços *Web*, existem três mecanismos básicos: provedores de serviços, registradores de serviços (*registry*) e consumidores de serviços. Provedores anunciam seus serviços no *registry*, que são usados pelos consumidores para localizarem e posteriormente utilizarem tais serviços.

Conceitualmente, Serviços *Web* são parecidos com plataformas ou *middewares* orientados a objetos. Nesses *middlewares*, objetos estão espalhados em diversos *sites* e interagem para comporem aplicações distribuídas, que, por sua vez, dispõem de serviços para o acesso remoto por parte de usuários. Localização, registro e comunicação entre os objetos distribuídos são realizadas por um *Object Request Broker* (ORB). Exemplos de *middlewares* para objetos distribuídos incluem CORBA da OMG [9], DCOM da Microsoft [6] e *Remote Method Invocation* (RMI) da *Sun Microsystems* [8]. Embora esses *middlewares* disponham da infra-estrutura básica para a construção de aplicações distribuídas, seu sucesso tem sido limitado por não utilizarem tecnologias amplamente dispiníveis na Internet. Por exemplo, na comunicação entre clientes e servidores em CORBA utiliza-se o protocolo *Internet Interoperable Protocol* (IIOP), o que impede a comunicação com servidores de corporações protegidas por *firewalls* (caso esses não tenham sido programados para receber trafego IIOP).

Serviços *Web* combinam as tecnologias *Web* e *middlewares* orientados a objetos numa infra-estrutura onde a interação entre componentes (ou objetos) e usuários é feita através de tecnologias padrões da *Web*, como o Extensible Markup Language (XML)[3] para representação dos dados e HTTP para comunicação. Dessa forma, um componente de um Serviço *Web* pode ser demandado por um *browser* ou por outro componente de outro Serviço *Web*. Uma vez publicados ou disponibilizados com identificadores universais da Web (*Uniform Resource Locators - URLs*), os serviços são acessados por protocolos padrões como o HTTP. Portanto, XML e HTTP são as tecnologias centrais de serviços *Web*, que são utilizados para comporem os demais mecanismos, listados a seguir:

- linguagem de descrição de Serviços (*WEB Service Description Language* - WSLD) - documento em XML usado para descrever a localização e interface provida por determinado Serviço;
- protocolo de troca de mensagens (*Simple Object Access Protocol* -SOAP) - protocolo baseado em XML para a comunicação entre provedores e consumidores de serviços; e,
- catálogo e localizador de serviços (*Universal Description, Discovery, and Integratio* - UDDI) - serviço baseado em XML e contém o registro dos Serviços *Web* e suas interfaces (i.e., formas de utilização).

1.2.2 Grades Computacionais (*grid computing*)

A idéia básica de uma grade (*grid*) é disponibilizar um ambiente com recursos computacionais abundantes (processamento e armazenamento) a partir do aproveitamento da integração de recursos mais modestos de vários computadores interligados em rede, de preferência via Internet. Idealmente, numa grade computacional, usuários e programas aplicativos podem compartilhar informações (bancos de dados e arquivos) e capacidade de processamento (processadores e memórias), sem a necessidade do conhecimento de onde esses recursos estão localizados. Ou seja, o usuário enxerga uma grade como um grande computador virtual, podendo este ter acesso seguro aos recursos do computador virtual a partir de qualquer dos computadores que formam a grade [27]. Essencialmente, não existem grandes novidades em relação aos problemas fundamentais que envolvem a construção de uma grade computacional, tais como compartilhamento de recursos, segurança, balanceamento de carga, acesso remoto e padrões abertos. Todos esses problemas têm sido discutidos e resolvidos no âmbito de pesquisas na área de sistemas distribuídos [1,2,3,5,7]. O que motiva o desenvolvimento de grades, na atualidade, é a possibilidade de estar usando a Internet para esse fim. Ou seja, a possibilidade de se estar usando

3 XML é uma linguagem de marcação de propósito geral cujo objetivo é facilitar o compartilhamento de dados na Internet entre sistemas heterogêneos (de fornecedores distintos).

a grande Rede como um supercomputador, da mesma forma que se usa o WWW como um repositório global de páginas da *Web*. Portanto, em um certo sentido, grade computacional e serviços *Web* são esforços complementares, visto que esses últimos podem ser utilizados como arcabouço de construção de determinada grade.

1.2.3 Sistemas *peer-to-peer*

O termo *peer-to-peer* se refere a sistemas totalmente distribuídos nos quais seus componentes são completamente equivalentes em termos das funcionalidades e tarefas que desempenham, embora algumas implementações *peer-to-peer* conhecidas (e.g., o Kazaa), usam os conceitos de servidores especiais com funções diferenciadas. De uma forma geral, como define Shirley [21], "*peer-to-peer* refere-se a uma classe de aplicações que tiram proveito de recursos disponíveis na Internet, como armazenamento, processadores, conteúdos e presença humana". Em um certo sentido, essas definições também se aplicam a várias aplicações de grades computacionais. No entanto, um aspecto de sistemas *peer-to-peer* que se distingue de grades computacionais é a capacidade que o sistema *peer-to-peer* tem de se adaptar a uma população transiente, se auto-organizando para incorporar e remover novos componentes no sistema de uma forma altamente dinâmica, sem a necessidade de uso de um componente ou autoridade central [13]. Tal característica dinâmica requer a adoção de topologias adaptáveis de redes e gestão de falhas (tolerância a falhas) de conexão de redes e dos computadores envolvidos. Existem sistemas *peer-to-peer* disponíveis para uma variedade de aplicações e serviços, como colaboração, banco de dados, distribuição de conteúdos, entre outras (*Chat*/Irc, Msn, PIER, Piazza, Gnutella, Groove, etc.) [13].

Grades computacionais e sistemas *peer-to-peer* têm objetivos semelhantes, ambos voltados para o compartilhamento de recursos distribuídos em larga escala, sendo que Grades, em geral, também se preocupam com o aspecto de alto desempenho das aplicações explorando a distribuição e execução paralela de tarefas.

1.2.4 Aglomerados (*Clusters*)

Um aglomerado ou *cluster* de computadores é um grupo de computadores ligados em rede (geralmente de alta velocidade) usados para melhorar a confiabilidade ou velocidade de aplicações e serviços. Um *cluster* de servidores *Web*, por exemplo, pode ser utilizado para aumentar a disponibilidade e desempenho, utilizando os princípios de balanceamento de carga e tolerância a falhas para permitir uma resposta rápida à solicitações de muitos clientes, mesmo que ocorram falhas de alguns dos componentes do *cluster* [22]. Um outro exemplo é a utilização de um *cluster* para produzir uma alternativa de custo razoável para os chamados supercomputadores. Um exemplo típico de *cluster* desse tipo é o *Beowulf*, originalmente desenvolvido pela NASA [23], mas que se tornou amplamente utilizado na comunidade científica como suporte à computação científica. Um *cluster Beowulf* é tipicamente composto por um conjunto de PCs idênticos, rodando sistemas do tipo UNIX de código aberto, como Linux ou BSD, e *software* de processamento paralelo como o *Message Passing Interface* (MPI) ou o *Parallel Virtual Machine* (PVM). Aplicações típicas de *clusters* vão desde previsão meteorológica (previsão do tempo e condições climáticas) até simulações financeiras.

O que distingue um *cluster* de outras tecnologias correlatas é o fato de que os computadores de um *cluster* são em geral homogêneos e dedicados, diferentemente do que ocorre nas grades computacionais e sistemas *peer-to-peer* que visam ambientes heterogêneos e abertos como a Internet.

1.2.5 Agentes Móveis

O que diferencia um agente móvel de um *software* convencional é sua capacidade de migrar autonomamente, ou mover-se de um computador para outro, a fim de continuar sua execução no computador destino. Ou seja, o agente formado pelo código do programa do agente e seus dados migra e continua a executar sem perder os estados da execução anterior. A principal vantagem de agentes móveis como forma de desenvolvimento de computações distribuídas é a redução do tráfego na rede, em contraposição ao que aconteceria numa arquitetura *client/server* convencional onde toda interação entre o cliente e servidor é feita através de troca de mensagens na rede. Sendo móvel, o agente fazendo o papel do cliente pode migrar para o computador onde se localiza o servidor e colher todos os dados necessários *in loco*.

Agentes Móveis têm sido usados para uma série de aplicações distribuídas como recuperação de informação e gerenciamento de redes [24] e requerem mecanismos de rastreamento de localização e coordenação sofisticados entre agentes [17].

2. PROJEÇÕES PARA OS PRÓXIMOS ANOS

Por agregar tecnologias conhecidas, como arquiteturas cliente/servidor e objetos distribuídos, com a possibilidade de uma integração global na Internet, os serviços *Web* são, nos dias de hoje, o principal foco de investimento para a construção das aplicações distribuídas. Mais ainda, várias agências de previsão estimam que os investimentos em serviços *Web* aumentarão significativamente nos próximos anos. Segundo o *International Data Corporation* (IDC), o gasto anual mundial em projetos de *software* baseados em serviços *Web* vai atingir a marca de 11 bilhões de dólares em 2008, contra 1,1 bilhão em 2003 [10]. Uma outra pesquisa conduzida pela agência Gartner em 2005 entre 110 empresas nos Estados Unidos, indicava que 54% dessas empresas já utilizavam serviços *Web* [10].

Se, de um lado, é inevitável a migração dos softwares distribuídos para as arquiteturas baseadas em serviços *Web* - de fato, muitas empresas e entidades governamentais já estão passando a desenvolver seus sistemas nesse novo ambiente [15] -, de outro lado, há uma série de desafios que, se não enfrentados adequadamente, poderão comprometer o uso efetivo dessa tecnologia em escala global. Entre esses desafios, se destaca a necessidade de segurança e privacidade, visto que um catálogo público de serviço (UDDI) é acessível a qualquer cliente conectado à Internet (algumas empresas certamente não desejariam expor detalhes de seus serviços para competidores). Da mesma forma, questões de garantias de integridade e autenticidade são vitais para evitar que agentes não autorizados modifiquem uma descrição de serviço (WSDL) ou mesmo introduzam código malicioso contendo, por exemplo, vírus. Soluções para esses problemas precisam seguir padrões abertos de modo a possibilitar que implementações de segurança de provedores de serviços distintos possam interoperar (i.e., sejam compatíveis). Para guiar a construção de soluções que atendam a esses requisitos, a IBM e a Microsoft publicaram um *white paper* [25]. Num outro documento, chamado de *Security Assertion Markup Language* (SAML), publicado pela Organização pelo Avanço de Padrões para Informação Estruturada (OASIS) e patrocinado por empresas, como Boeing, Citrix, HP, IBM, Microsoft, Netscape e Sun Microsystems, foi publicado um padrão aberto para especificação de segurança que permitirá a interoperação de soluções de segurança de provedores distintos de serviços *Web* [26].

Diante da onipresença dos serviços *Web*, nos mais variados tipos de aplicação, um outro desafio importante para desenvolvedores é o chamado *front-end integration*, que tem de levar em conta a necessidade de integração da multiplicidade de dispositivos disponíveis para usuários de serviços na Internet, que incluem *laptops*, *browsers*, *Personal Digital Assistant* (PDAs) e telefones celulares.

Serviço continuado, sem interrupções, é outro fator importante para uma classe de aplicações chamadas críticas. De acordo com Ken Birman [11], existe uma tendência cada vez maior de empresas e governos usarem serviços *Web* em aplicações de caráter crítico, que poderiam causar grandes prejuízos caso sofressem interrupções em seus serviços. Entre essas aplicações, encontram-se serviços bancários, serviços *online* de companhias, sistemas de automação e controle industrial [16] e sistemas que operam infra-estruturas vitais a exemplo de energia elétrica e transporte. Ainda segundo Birman, apesar de existirem padrões para confiabilidade para serviços *Web* (WS-*Reliability* e WS-*Transactions*), esses apresentam limitações, pois não abordam o problema do serviço continuado sem interrupções, mesmo na presença de falhas acidentais, intencionais ou reconfigurações dos sistemas. Por exemplo, qual seria a conseqüência de um ataque de vírus ou de um ataque do tipo negação de serviço (*denial of service*) numa planta nuclear ou num sistema de negociação de ações? Por outro lado, os mecanismos para resolver esses tipos de problemas já foram desenvolvidos pela comunidade de computação distribuída, como por exemplo, os mecanismos de replicação ativa, comunicação em grupo e detecção de intrusão [1,2]. Restando apenas que essas soluções conhecidas sejam também adotadas nos padrões de serviços *Web* - o que ainda está para ser feito.

Um outro aspecto não menos importante é o aumento do trafego na Internet causado pelo uso extensivo de XML. Uma previsão feita pela firma ZapThink prevê que em 2006 XML representará 26% de todo tráfego na Internet mundial e 40% em 2008 [12]. A preocupação nesses números decorre do fato de que boa parte de uma mensagem em XML representa meta-dados e não a informação a ser transmitida propriamente dita. Para contornar esse problema, o *World Wide Consortium* (W3C) conjuntamente com a *Sun Microsystems* estão desenvolvendo um formato padrão para uma versão mais econômica do XML chamado XML binário [12].

Em suma, diante da inevitável adoção dos serviços *Web* num mundo globalizado, onde aplicações distribuídas passam a ser construídas a partir da composição de serviços que podem estar dispersos em diversas partes do mundo, demandando diferentes contratos de utilização, abrem-se novas oportunidades para provedores de serviços, consumidores de serviços, negociadores de serviços, integradores de componentes, e enfim, negócios em geral que incluam serviços *Web* como parte de seus processos. No entanto, serviços Web trazem consigo novos desafios tecnológicos a serem vencidos agora e no futuro, que demandarão esforços da academia, indústria de *software* e desenvolvedores em geral.

3. POLÍTICAS REFORÇADORAS

Diante dos desafios que se apresentam com a rápida disseminação da computação distribuída, grandes oportunidades se seguirão no mercado globalizado, mas que somente se concretizarão nos seguimentos sociais onde existam os conhecimentos das tecnologias de computação distribuída subjacentes e a capacidade de inovação tecnológica. Para atingir esses objetivos plenamente, algumas ações se fazem necessárias, onde se destacam:

1. Intensificação na formação de mão de obra qualificada em nível de graduação e pós-graduação na área de ciência da computação, com foco nas tecnologias de computação distribuída. Nesse sentido cabem ações de Governo de apoio a projetos didáticos que contemplem esforços nessa área;
2. Identificação de áreas estratégicas, no governo e sociedade em geral, onde as tecnologias de computação distribuída possam potencializar maior desenvolvimento (e.g., segurança pública, saúde, gestão governamental, indústria, etc.);
3. Indução de cooperações entre empresas e academia para viabilizar as inovações tecnológica desejáveis. Essas induções poderiam ser concretizadas pelo Governo através de chamadas de projetos de desenvolvimento tecnológico focados em temas identificados no item 2.

4. CONCLUSÕES

A Computação Distribuída, cujo desenvolvimento científico iniciou-se ainda na década de setenta, tornou-se uma realidade global somente depois do advento da Internet e da WWW. Os Serviços *Web* foram desenvolvidos com o intuito de criar uma infra-estrutura capaz de integrar as iniciativas das várias industrias de Tecnologia da Informação (TI) e demais pólos de desenvolvimento, e passou a ser o grande foco mundial para o desenvolvimento de novas aplicações e integração com sistemas distribuídos legados. Paralelamente aos serviços *Web*, várias tecnologias de desenvolvimento de Computação Distribuída surgiram com o objetivo de facilitar o desenvolvimento e aproveitar recursos computacionais dispersos, onde se destacam Grades Computacionais, sistemas *Peer-to-Peer*, *Clusters* e Agentes Móveis. Por outro lado, essas tecnologias tendem a se relacionar de diversas formas. Por exemplo, agentes móveis podem ser utilizados na construção de grades e pode-se ter grades para sistemas *peer-to-peer*. Igualmente, serviços *Web* também podem ser utilizados para implementar essas tecnologias e essa tem sido uma tendência na área.

Com a enorme quantidade de computadores disponíveis na Internet, a exploração de computações distribuídas e em particular, estruturadas em serviços *Web* parece ser um caminho natural, abrindo imensas oportunidades para desenvolvedores de TI e serviços. Entretanto, para o pleno sucesso desse novo paradigma, vários desafios se apresentam, onde se destacam: questões de segurança - como a eliminação ou controle de ataques epidêmicos (vírus, *worms*, spams, negação de serviço) -, a necessidade de integração de aplicações e serviços, a necessidade de serviço continuado mesmo na presença de falhas (i.e., tolerância a falhas), a usabilidade em termos de qualidade de serviço (QoS) e, finalmente, novas leias e regras para estabelecimento de contratos de serviços entre provedores e consumidores.

Por fim, o domínio dos conhecimentos subjacentes à Computação Distribuída e seu aproveitamento no mercado global emergente requerem ações pontuais em Educação, Ciência e Tecnologia, que também apontamos neste artigo.

5. REFERÊNCIAS

[1] Mullender S. (Editor), "Distributed Systems", ACM Press, New-York, 1993.

[2] Lynch N.,"Distributed Algorithms", Morgan Kaufmann Pub., San Francisco (CA), 1996.

[3] Lamport, L., "Time, clocks, and ordering of events in a distributed system", Communication ACM, 21, 7 (July 1978), pp. 558-565.

[4] M. Fischer, N. Lynch, and M. Peterson, "Impossibility of Distributed Consensus with One Faulty Process", J. ACM, 32, April 1985, pp 374-382.

[5] Chandra, T. and Toueg, S, "Unreliable Failure Detectors for Reliable Distributed Systems", Journal of ACM, 43(2), pp. 225-267, March 1996.

[6] DCOM Specifications, url: http://www.microsoft.com/oledev/olecom/draft-brown-dcom-v1- spec-02.txt/

[7] Hurfin, M., Macêdo, R., Raynal, M., and Tronel, F., "A General Framework to Solve Agreement Problems", Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems (SRDS'99), Lausanne, Switzerland, pp. 56-67, October/1999.

[8] JAVA Specifications, url: http://java.com.sun/

[9] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, OMG. Feb. 1998, url: http://www.omg.org/

[10] Leavitt, N., "Are Web Services Finally Ready to Deliver?", IEEE Computer, Volume 37, Number 11, November 2004, pp. 11-18.

[11] Birman, K., "Can Web Services Scale Up?", IEEE Computer, Volume 38, Number 10, October 2005, pp. 107-110.

[12] Geer, D., "Will Binary XML Speed Network Traffic?", IEEE Computer, Volume 38, Number 4, April 2005, pp. 16-18.

[13] Androutsellis-Theotokis, A. and Spinnellis, A., "A Survey of Peer-to-Peer Content Distribution Technologies", ACM Comp. Surveys, V. 36, N. 4 , December 2004, pp. 335- 371.

[14] Huhns, M. and Singh, M. P., "Service-Oriented Computing: Key Concepts and Principles", IEEE Internet Computing, Jan-Feb 2005, pp. 75-81.

[15] Umar, A., "The Emerging Role of the Web for Enterprise Applications and ASPs", Proceedings of the IEEE, Volume 92, Number 9, September 2004, pp. 1420-1438.

[16] Andrade, S. and Macêdo, R., "A Component-Based Real-Time Architecture for Distributed Supervision and Control Applications", Proceedings of the 10 th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005), p. 15-22, Italy, Sept/2005.

[17] Macêdo, R. and Silva, F., "The mobile groups approach for the coordination of mobile agents", Journal of Parallel and Distributed Computing, Elsevier, v. 65, n. 3, p. 275-288, 2005.

[18] Mullender, S, Rossum, G, Tanenbaum, A., Renesse, R., and Stareven, J., "AMOEBA A Distributed Operating System for the 1990s", IEEE Computer 23(5).

[19] Magee, N., Dulay, N., and Kramer, J., "Structuring Parallel and Distributed Programs", Software Eng. Journal, Vol.8, No 2, pp. 73-82, 1993.

[20] [url: http://www.w3.org/2002/ws/

[21] Shirky, C., "What is p2p... and what isnt't", Network, disponível online em http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html/, O'Reilly. 2000.

[22] D. Ingham, S. Shrivastava, P. Panzieri. "Constructing Dependable Web Servers", IEEE Internet Computing, V. 4, N. 1, Jan/2000.

[23] url: http://www.beowulf.org/

[24] Fuggetta, A., Picco, G.P., and Vigna, G., "Understanding code mobility", IEEE Trans. Software Eng. 24 (5), May/1998.

[25] url: http://msdn.microsoft.com/library/en-us/dnwssecur/securitywhitepaper.asp

[26] url: http://www.oasis-open.org/SAML

[27] Berman F., Hey A. J.,and , Fox G., "Grid Computing: Making The Global Infrastructure a Reality", Wiley. url: http://www.grid2002.org/.

Raimundo José de Araújo Macêdo é Professor Titular do Departamento de Ciência da Computação da UFBA; Ph.D. em Ciência da Computação pela *University of Newcastle upon Tyne*, Inglaterra; Coordenador do Laboratório de Sistemas Distribuídos (LaSiD) e do Programa de Pós-Graduação em Mecatrônica da UFBA; foi presidente da Comissão Especial em Redes de Computadores e Sistemas Distribuídos da Sociedade Brasileira de Computação (1999-2001) e Coordenador Geral e do Comitê de Programa do Simpósio Brasileiro de Redes de Computadores (1999 e 2002).