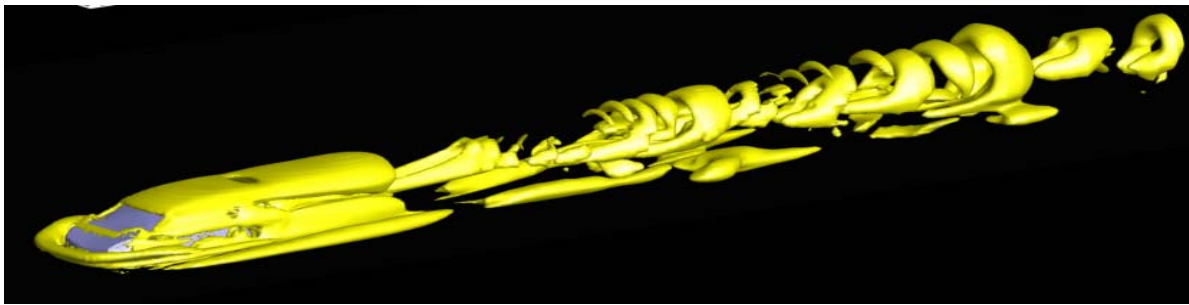
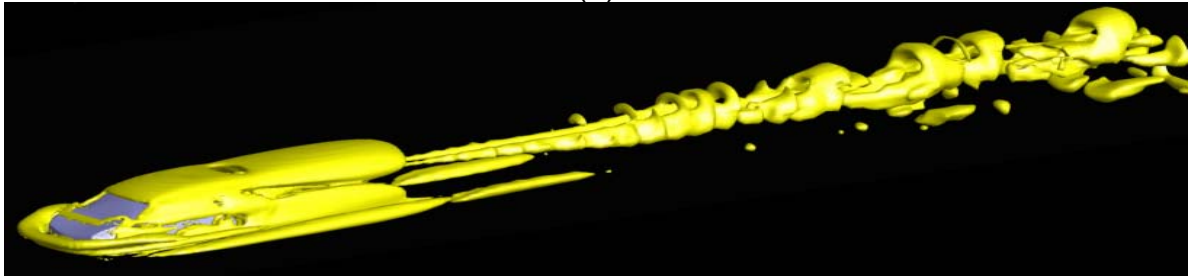




(e)



(f)

Figura 4.34. Evolução temporal de isosuperfícies de $Q = 5$ para os tempos físicos: (a) $t = 0,05s$, (b) $t = 0,045s$, (c) $t = 0,09s$, (d) $t = 0,8s$, (e) $t = 2,7s$ e (f) $t = 4,7s$ (Vedovoto, 2007).

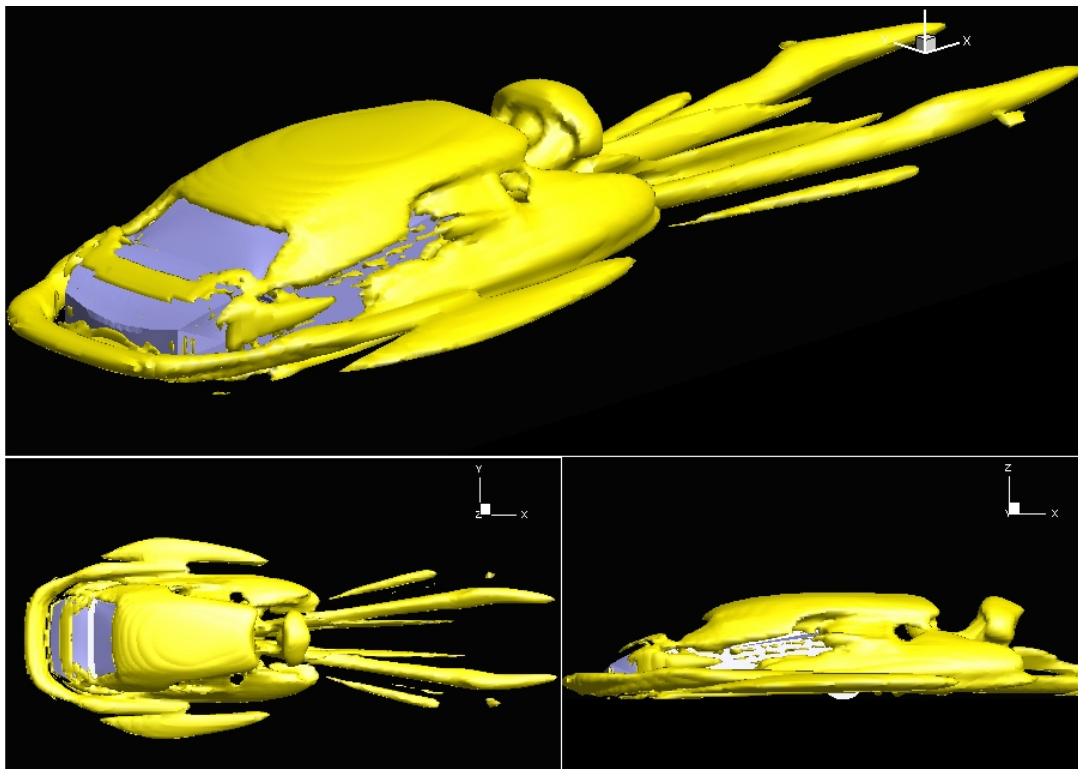


figura 4.35. Detalhes das isosuperfícies de $Q = 10$ para $t = 0,6s$ (Vedovoto, 2007).

Realizando uma análise comparativa entre o escoamento obtido com a malha do lamborghini (Vedovoto, 2007), em relação ao escoamento sobre um cubo, pode-se notar a semelhança das estruturas turbilhonares, mesmo com condições de simulação diferentes.

Hwang e Yang, 2004, estudaram o escoamento sobre um cubo solidário a uma parede em $Z = 0$. A figura 4.36 mostra isosuperfícies de Q obtidos por Hwang e Yang (2004).

Tanto na figura 4.35 quanto na figura 4.36, é possível visualizar claramente, os vórtices em formato de ferradura, vórtices laterais e vórtices em formato de grampo de cabelo. Com esse tipo de comparação é possível confirmar a viabilidade das malhas geradas pelo GMSH, mesmo sendo as malhas geradas por composição de faces, resultando em irregularidades na união das mesmas.

É válido ressaltar que, a boa qualidade das malhas complexas, depende da habilidade ao desenhar as geometrias. Sendo assim, para que os resultados quantitativos possam ser validados, eles dependerão da virtualização do protótipo, e da riqueza de detalhes do mesmo.

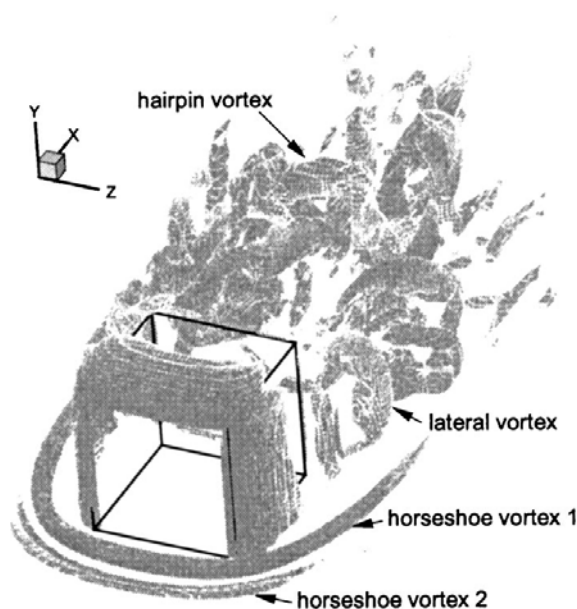


Figura 4.36. Isosuperfícies de Q obtidos por Hwang e Yang (2004), $Re = 1.000$.

4.3.2. Protótipo de uma aeronave

A geometria do protótipo em questão foi baseada em uma aeronave comercial (Legacy 600), de fabricação da EMBRAER. Apesar de ser composta por geometrias circulares, a complexidade na virtualização de seu protótipo é bem mais elevada que a do lamborghini Gallardo, pois o lamborghini teve sua estrutura criada em paralelo ao eixo x . Já o Legacy, como mostrado na figura 4.37, possui um ângulo de inclinação em relação ao eixo x .

Caso o avião fosse projetado em paralelo ao eixo x , todas as circunferências que compõe sua estrutura sofreriam modificações apenas na coordenada ao longo do eixo x .

Porém, com a inclinação acima citada, cada ponto das circunferências têm que ser recalculados à medida que se variam as posições em x, como mostrado na figura 4.38 abaixo.

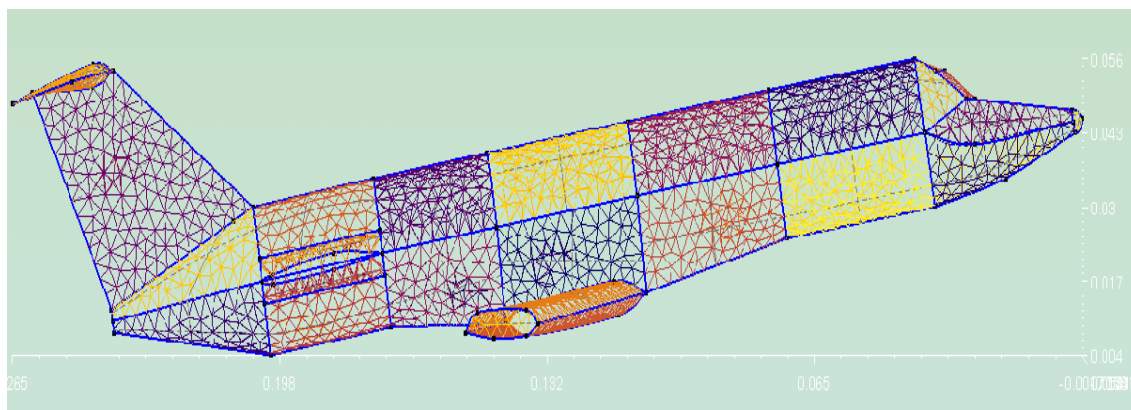
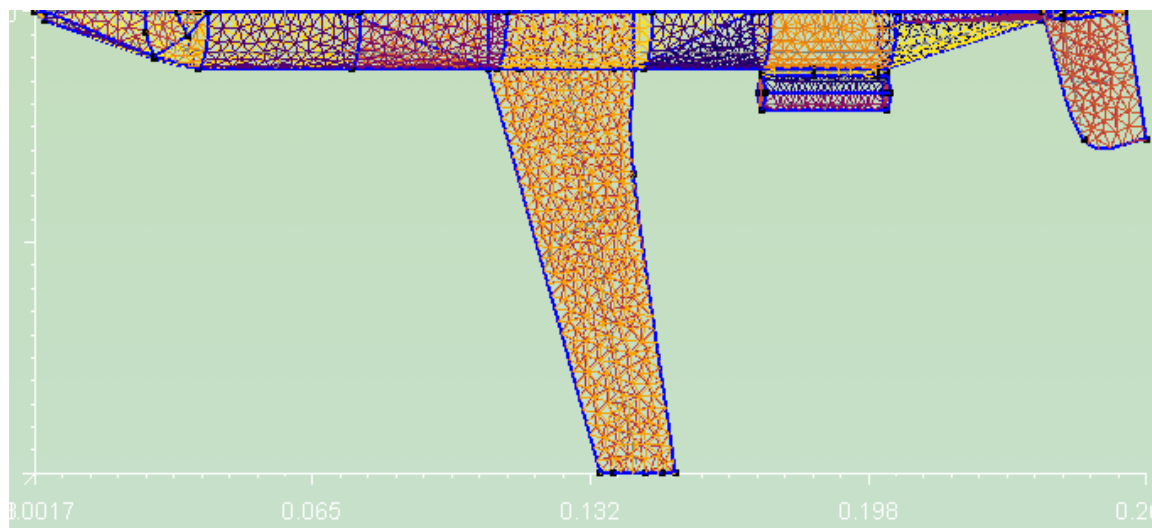


Figura 4.37. Malha do Protótipo do Legacy 600 com 10° de inclinação em relação ao eixo x.

Deve-se lembrar que a metodologia utilizada na virtualização deste protótipo, seguiu o mesmo padrão utilizado na criação do lamborghini, partindo da criação do perfil da aeronave. Como o modelo possui estrutura tubular, o perfil do mesmo é referencial da divisão da aeronave ao meio exato, como mostrado na figura 4.38, através do qual se cria a simetria entre as partes para produção da geometria completa.



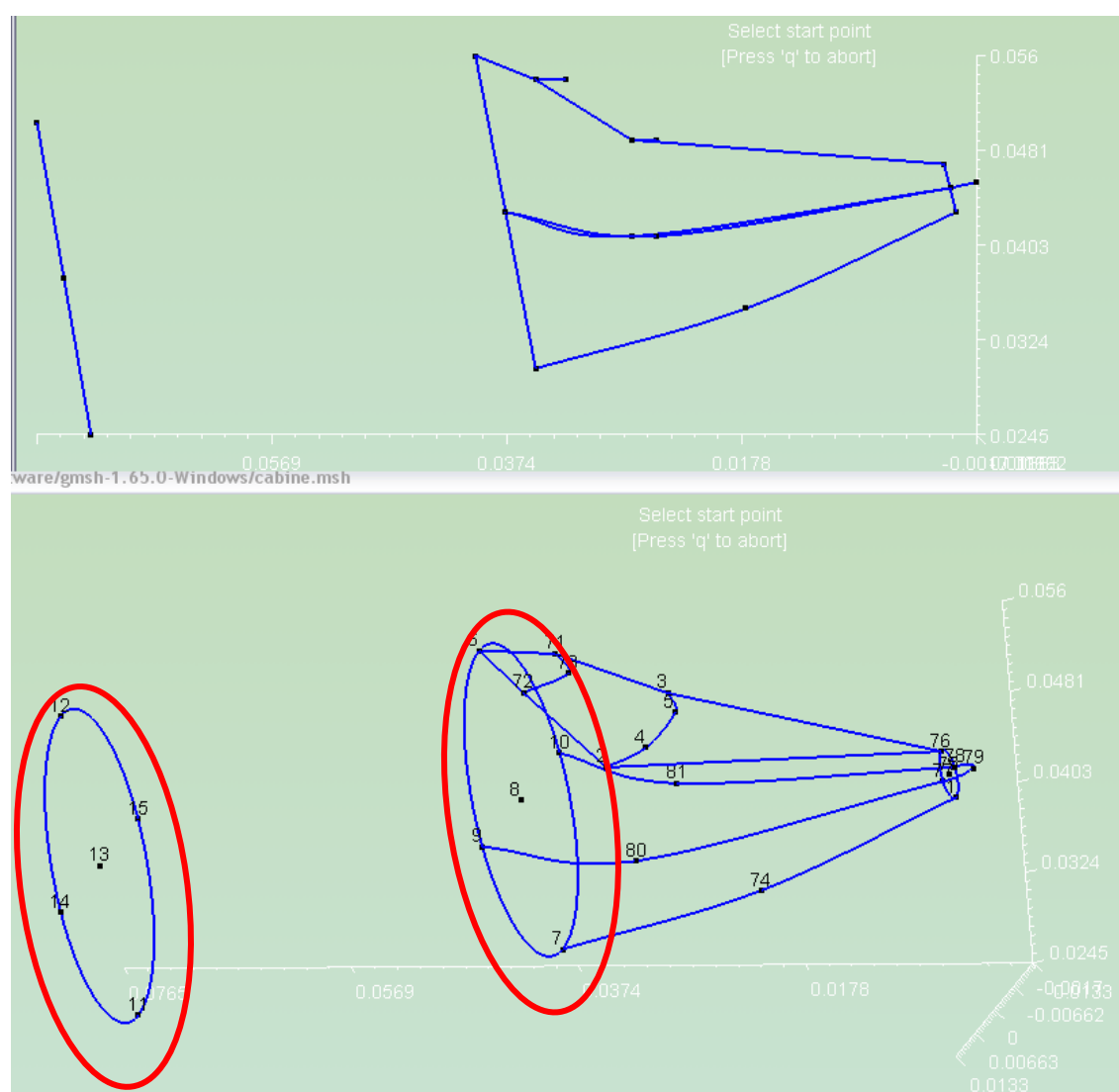


Figura 4.39. Detalhe dos pontos da geometria da cabine do avião.

Com essa inclinação (figura 4.39), pode-se notar a mudança em todos os pontos de cada circunferência, dando destaque para as que estão circuladas em vermelho, sendo os pontos mostrados na figura 4.40 abaixo, e essas mudanças persistem até o fim da virtualização do projeto.



Figura 4.40. Variação das coordenadas dos pontos da geometria.

A malha de elementos triangulares que representam o protótipo pode ser visualizada com mais detalhes na figura 4.41 abaixo.

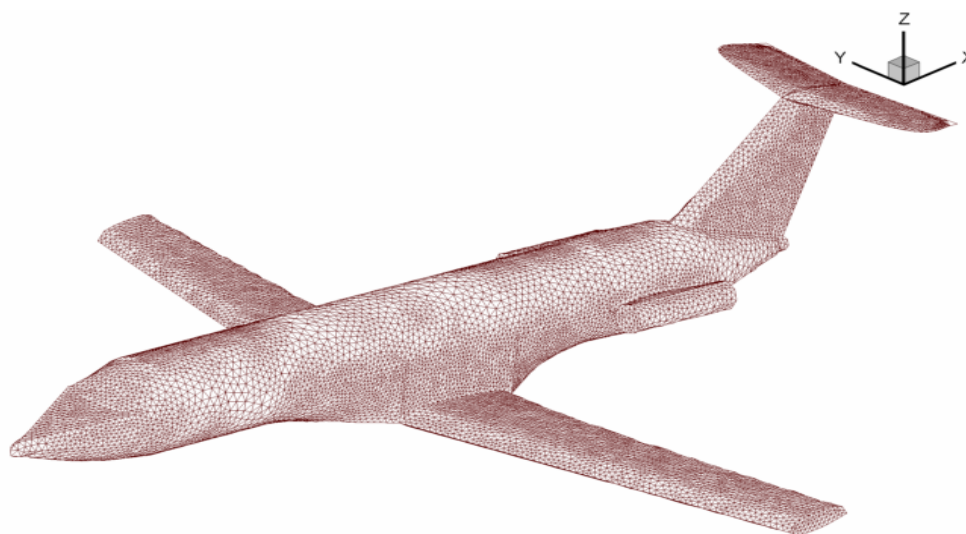


Figura 4.41. Malha de elementos triangulares do protótipo de aeronave.

Este protótipo virtual pode ser melhor detalhado com um processo de virtualização mais rigoroso, de forma que seja rico em detalhes. Um exemplo de detalhe a ser acrescentado para dar riqueza ao projeto, é mostrado na figura 4.42 abaixo, com a virtualização do trem de pouso do avião.

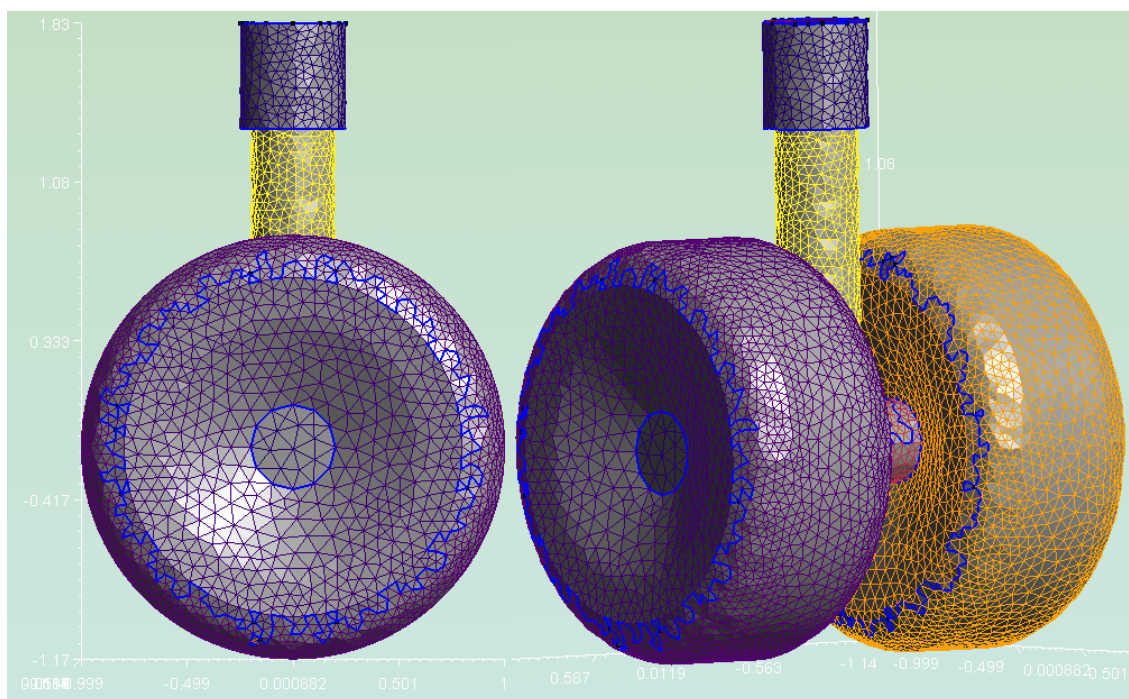


Figura 4.42. Detalhes de um trem de pouso de avião.

De acordo com a visualização gerada por Vedovoto (2007), a figura 4.43 abaixo, mostra algumas linhas de corrente posicionadas sobre o protótipo de aeronave, gerada neste projeto. Uma vez que um dos objetivos deste trabalho é verificar a capacidade de importação (upload) de geometrias complexas, não houve ainda uma preocupação de se simular o protótipo da aeronave a altos números de Reynolds, ou realizar a simulação em domínios com razões de aspecto muito grandes.

Nota-se que apesar do domínio escolhido não ser totalmente adequado (o tamanho característico da malha euleriana na região da fronteira imersa é duas vezes maior que o tamanho característico da malha lagrangiana), o código convergiu e mostrou resultados fisicamente consistentes. Portanto, o que se pode observar mediante as malhas geradas pelo GMSH, que é possível obter resultados quantitativos de forma a realizar, por exemplo, otimização de estruturas, através da virtualização de geometrias e da simulação numérica do escoamento.

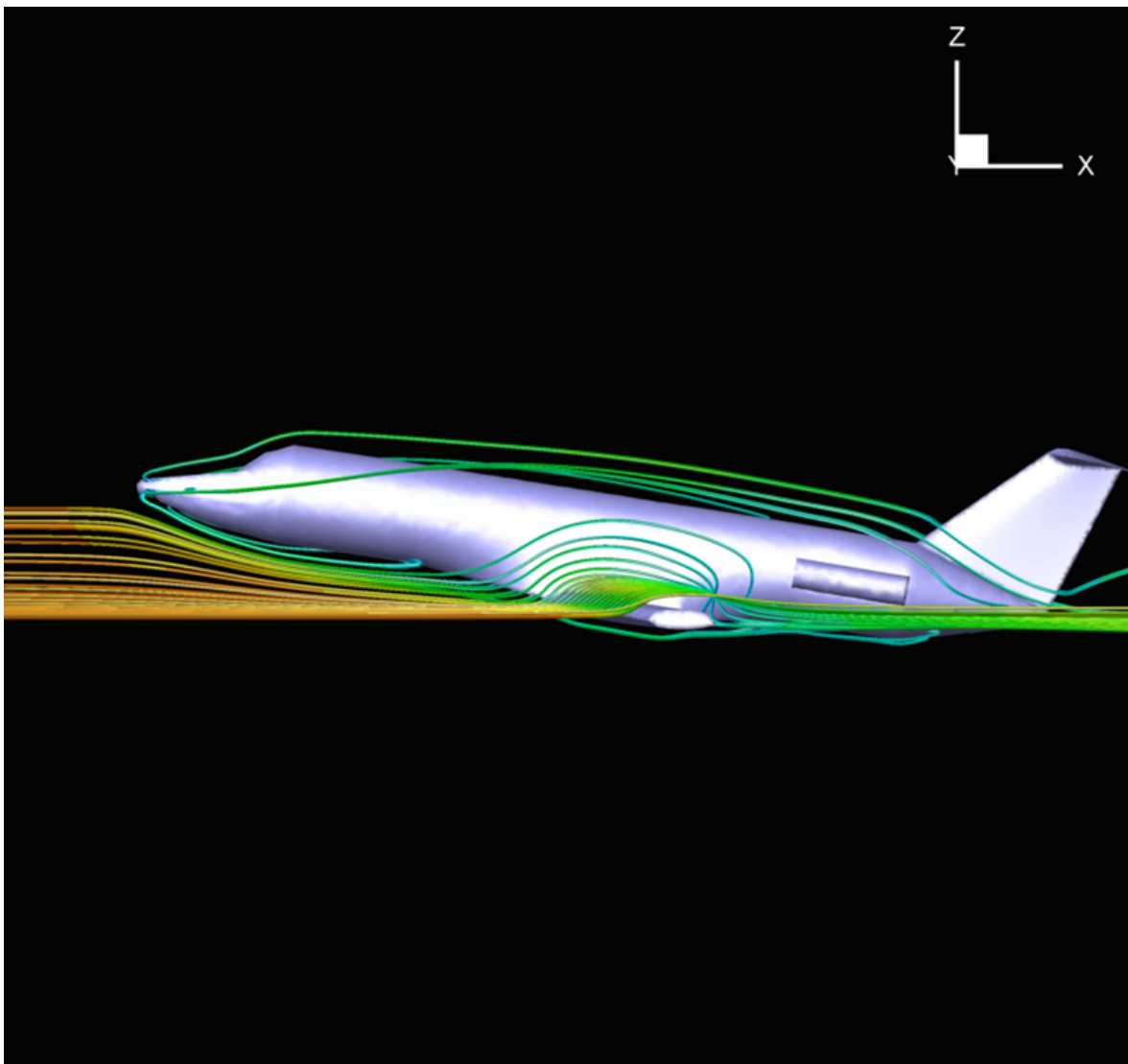


Figura 4.43. Linhas de corrente indicando o padrão de escoamento sobre o protótipo da aeronave (Vedovoto, 2007).

CAPITULO 5

Conclusão e Trabalhos Futuros

A aplicação da proposta metodológica apresentada permitiu auxiliar no processo de desenvolvimento da interface para o solver do LTCM, em forma de *site*, em um procedimento de virtualização de protótipos e no processo de geração de malhas superficiais.

As fases da metodologia tornaram este trabalho possível, pois na fase inicial resumiu-se a entrada de dados, através da coleta de informações sobre o código existente no LTCM e informações obtidas com os usuários do código, orientando o processo de delimitação dos elementos visuais presentes na interface segundo os objetivos de um *site* eficiente.

Embora a complexidade do projeto esteja diretamente ligada ao tipo de *site* a ser criado (portal, pessoal, institucional, comercial, educacional, entre outros), este fato não foi o fator delimitador deste projeto.

No presente trabalho propôs-se elaborar uma interface gráfica que pudesse ser aplicada ao solver de CFD desenvolvido no LTCM, com a finalidade de atrair usuários ao uso do solver, oferecendo facilidade de interação homem-software. É importante salientar que uma interface em forma de *site*, produzido com a finalidade de viabilizar a comunicação entre usuários e o solver, não é mais importante que uma interface gráfica de instalação local.

Com a implementação da interface, percebe-se que mesmo sendo a iteração homes/software via Web ou de instalação local, ambas têm objetivos comuns. Cada uma possui suas qualidades e facilidades no manuseio, porém o processo metodológico para criação de um *site*, amplia o acesso às informações, pois as informações são manipuladas e viabilizadas de qualquer local com acesso a internet.

Percebeu-se que trabalhar com conhecimentos específicos, tornando-os requisitos técnicos do processo projetual foi uma tarefa enriquecedora. No que diz respeito a vincular as informações específicas de mecânica dos fluidos de forma prática e objetiva, oferecendo os conteúdos mais básicos e necessários para a interface, constatou-se que, pelo fato do código computacional já estar parcialmente validado, a maior preocupação foi evitar a redundância de informações a serem enviadas para o solver do LTCM. Verificando-se esta preocupação, viabilizou-se uma solução, ainda que simples, sobre o conceito de interface gráfica, porém funcional e versátil.

O presente trabalho alcançou os objetivos propostos, limitando-se ao enfoque do tema sob a ótica da comunicação entre homem/software, o que reforça a não presença de orientação e suporte de conteúdos sobre alteração ou desenvolvimento do código computacional existente. Com isso, pode-se visualizar que a metodologia de implementação para a interface do Fluids-3D possui caráter generalista, deixando possibilidades de melhoras na interface, tais como módulos adicionais, à medida que os mesmos sejam acrescentados ao solver.

Prevê-se que futuramente a WEB esteja viável a um grande grupo de pessoas através da televisão, o que já se constata, ainda que de forma incipiente, através de TVs via cabo ou satélite. Por esse motivo, supõe-se que a comunicação continuará sendo o objetivo dos *sites*, porém as inovações tecnológicas irão delimitar novas possibilidades que poderão afetar os conhecimentos básicos necessários ao profissional. Isso nos leva a crer que, o conhecimento, seja ele geral ou específico como o de mecânica dos fluidos, poderá ser disponibilizado de maneira rápida e funcional.

Além dos resultados alcançados com a produção da interface gráfica acima citada, no presente trabalho, dedicou-se a busca por tecnologias gratuitas na área de geração de malhas de geometrias complexas.

O uso das ferramentas para criar a interface e viabilizar a comunicação da interface com o solver, permitiu selecionar uma ferramenta gratuita de geração de malha, chamada GMSH, a qual possui uma funcionalidade satisfatória para as necessidades do projeto.

É importante salientar que, o fato de valer-se de ferramentas gratuitas na composição de trabalhos científicos, exige uma necessidade maior de se buscar resultados comparativos, para validar as informações computadas.

Mesmo obtendo bons resultados com a ferramenta geradora de malha escolhida, foi notório que softwares livres possuem suas limitações, como dificuldades em se encontrar documentações, existência de erros (bugs), etc; problemas esses que não foram impecinhos no desenvolvimento do projeto.

Como trabalhos futuros, deixa-se as seguintes recomendações que podem melhorar

ainda mais a interface desenvolvida:

Implementar o gerenciamento de usuários da interface, através de filtros que possam restringir o número de usuários e o número de máquinas que os mesmos possam acessar no cluster.

Acrescentar à interface, módulos de pós-processamento, tal qual usado pelo Ganglia, por exemplo, que obtém informações quantitativas e as transformam em gráficos, através do uso da linguagem PHP com a ferramenta RRDtool.

Disponibilizar uma ferramenta geradora de malha através da Internet, vinculada à interface, de forma a diminuir uma etapa na simulação de casos de mecânica dos fluidos; além de possibilitar ao solver a capacidade de importação de diferentes listas de geometrias;

Incluir um módulo de visualização de arquivos no formato STL, o qual pode ser feito em linguagem Java e inseridos no código PHP da interface, dando maior caráter gráfico a interface em questão.

CAPITULO 6

Referências Bibliográficas

ARAÚJO, A. C. Framework de Análise e Projeto Baseado no RUP para o desenvolvimento de aplicações web. Dissertação de mestrado publicado na UFPE, Recife, 2001

BARRETT, E. MIT guide to teaching web *site* design. Cambridge: Mit Press 2001

BARRETT, E. The Society of text: hypertext, hypermedia, and the social construction of information. Series: Information systems. Cambridge: Mit Press, 1989.

BLACK, R. *Web Sites that work*. San Jose:Adobe Press, 1997.

BITTNAR, Z.; RYPL, D. ``*Direct Triangulation of 3D Surfaces Using the Advancing Front Technique*”, In: Numerical Methods in Engineering '96 (ECCOMAS '96), Wiley & Sons, 86-99, 1996.

BONSIEPE, G. *Design – do material ao digital*. Florianópolis, FIES/IEL, 1997.

BÜRDEK, B. E. *Diseño – História , teoria y práctica del diseño industrial*. Barcelona: Gustavo Gili, 2ª ed., 1999.

CAMPREGHER, R. “Modelagem Matemática Tridimensional Para Problemas de Iteração Fluido-Estrutura”. Tese de Doutorado. Universidade Federal de Uberlândia, 2005.

DALHEIMER, M., “Programação em Qt”, Editora Moderna, 1999;

DYN, N.; GREGORY, J.A.; LEVIN, A. ``*A Four-Point Interpolatory Subdivision Scheme for Curve Design*”, Computer Aided Geometric Design, Vol. 4, 257-268, 1987.

GATTASS, M.; LEVY., C. H. New Techniques to Aid Interactive Graphical Finite Element Analysis, Anais do Workshop Internacional de Aplicação de Mecânica Computacional em Geotecnia, Rio de Janeiro, 29-31 de junho de 1991.

GILMANOV, A.; SOTIROPOULOS, F; BALARAS, E. "A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids", Journal of Computational Physics 191, 660-669, 2003.

GLENN, B. T.; CHIGNELL, M.H.; Hypermedia: Design for Browsing. In: Hartson, HR Hix, D. Advances in Human-Computer Interaction. Nerwood, New Jersey: Ablex Publishing Corporation, v.3, 1992

HALLER, G., 2005, "An objective definition of a vortex", Journal of Fluid Mechanics 525, 1-26.

HALSTEAD, M.; KASS, M.; DEROSE, T. *"Efficient, Fair Interpolation using Catmull-Clark Surfaces"*, In: Computer Graphics Proceedings (SIGGRAPH '93), 35-44, 1993.

HARTSON, H. R.; HIX, D. Human-Computer Interface Development: Concepts and Systems for its Management, ACM Computing Surveys (21), No. 1, pp. 9-93, March 1990.

HWANG, J., YANG, K. "Numerical study of vertical structures around a wall-mounted cubic obstacle in channel flow", Physics of Fluids, Vol. 16, No. 7, 2382-2394, 2004.

JEONG, J. E HUSSAIN, F. "On the identification of a vortex", Journal of Fluid Mechanics 285,69-94, 1995.

LIMA E SILVA, A. L. F., 2002, "Desenvolvimento e implementação de uma nova metodologia para modelagem de escoamentos sobre geometrias complexas: método da fronteira imersa como Modelo Físico Virtual", Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

MARCUS, A., VAN DAM, A. ; User-interface developments for the nineties, 1991.

MARTINS, V. ; O processo Unificado de Desenvolvimento de Software, Companhia de Informática do Paraná – CELEPAR: Paraná, 1999. Disponível em: <http://www.pr.gov.br/celepar/batebyte/edicoes/1999/bb89/software.htm>. Acessado em : 14/06/2003

MOK, C. *Designing business – multiple media, multiple disciplines*. São Jose: Adobe Press, 1996.

MUTO, C. A., PHP & MySQL: Guia Introductório. Rio de Janeiro: Editora Brasport, 2006.

NIEDERAUER, J. PHP 5, Editora Novatec, Editora Brasport. 2005.

OLIVEIRA, G. P.; VEDOVOTO, J.M.; FERNANDES, A.P.; NETO, A.S. Desenvolvimento de uma interface gráfica para dinâmica dos fluidos computacional, utilizando a metodologia de fronteira imersa, In proceedings of 11th Brazilian Congress of Thermal Sciences and Engineering – ENCIT, 2006.

PERUZZI, J. T. *Manual sobre a importância do design no desenvolvimento de produtos*. Bento Gonçalves: SENAI/CETEMO/SEBRAE, 1998.

PETERS, R., The Simplest Subdivision Scheme for Smoothing Polyhedra - ACM Transactions on Graphics 16(4):420-431, 1997.

PITANGA, M, 2002, “Construindo supercomputadores com Linux”, Editora Brasport, 2004.

RADFAHRER, L.. *Design/Web/Design*. São Paulo: MarketPress, 1999.

RAY, J. Aprendendo em 21 dias Dreamweaver UltraDeb 4, Editora Campus, 2001.

RYPL, D.; BITTNER Z. “*Triangulation of 3D Surfaces: From Parametric to Discrete Surfaces*”, In: CD-ROM Proceedings of the Sixth International Conference on Engineering Computational Technology, Civil-Comp Press, 2002.

RYPL, D.; BITTNER Z. “*Triangulation of 3D Surfaces Reconstructed by Interpolating Subdivision*”, to appear in Computers and Structures.

SHAMES, I. H. Mecânica dos Fluidos. Volume 1. Princípios Básicos. São Paulo: Editora Edgard Blucher Ltda., 1973.

STERNE, J. Marketing na Web: Integrando a Web à Sua Estratégia de Marketing. São Paulo: Campus, 2000.

STOLLNITZ, E.J.; DEROSE, T.D.; SALESIN, D.H. *Wavelets for Computer Graphics: Theory and Applications*, Morgan Kaufmann Publishers, Inc., 1996.

TOUSSAINT, G. T.; Computational Geometry on the web. Disponível em: <<http://cgm.cs.mcgill.ca/~godfried/teaching/cg-web.html>>

UNVERDI, O.; TRYGGVASON, S. Computations of multi-fluid flows. *Physica D: Nonlinear Phenomena*, 1992.

VEDOVOTO, J. M. Modelagem Matemática e Simulação Numerica de Escoamentos sobre Geometrias Complexas Tridimensionais Utilizando o Método da Fronteira Imersa e Processamento Paralelo, Dissertação em redação, Universidade Federal de Uberlândia, Uberlândia, 2007.

VEDOVOTO, J.M.; CAMPREGHER, R.; SILVEIRA NETO, A. "Mathematical modeling and numerical simulation of a three-dimensional flow over complex geometries using the immersed boundary method", In proceedings of 11th Brazilian Congress of Thermal Sciences and Engineering – ENCIT, 2006.

ZORIN, D.; SCHRÖDER, P.; SWELDENS, W. *Interpolating Subdivision for Meshes with Arbitrary Topology*, In: Computer Graphics Proceedings (SIGGRAPH '96), 189-192, 1996.

VIDA DIGITAL. Parte integral da Revista Veja. São Paulo: Abril, Dezembro 2001. Ano 33, nº 52.

<http://pt.wikipedia.org/wiki/CFD> acessado em: 20/11/2006.

GEUZAINÉ, C., et al. "Gmsh Reference Manual: The documentation for Gmsh 1.60", 2005. Disponível em: <<http://www.geuz.org/gmsh/>> Acessado dia: 20/08/2005;

SCHÖBERL, J., "Netgen – 4.3", 2003. Disponível em: <<http://www.hpfem.jku.at/netgen>> Acessado dia: 23/08/2005;

<<http://www.cs.cmu.edu/~quake/triangle.html>> Acessado dia: 02/09/2005;

<<http://www.borland.com.br/delphi>> Acessado dia 05/10/2005;

<<http://www.borland.com.br/kylix>> Acessado dia 05/10/2005;

<<http://www.trolltech.com/qt/designer.html>> Acessado dia 08/10/2005;

<<http://doc.trolltech.com/3.3/designer-manual.html>> Acessado dia: 08/10/2005;

Anexo 1. Tutorial do GMSH

O GMSH é um software gerador de malhas não-estruturadas para métodos numéricos (MEF – Métodos de Elementos Finitos), incluindo uma ferramenta CAD para desenho de geometrias 2D e 3D. O GmsH conta ainda com uma suíte de visualização de dados para pós-processamento de resultados, baseado em OPEN GL.

O GMSH possui uma organização constituída de 4 módulos: Geometry, Mesh, Solver e Post-processing como mostrado na figura 1 abaixo, além dessa organização em módulos, ele ainda apresenta uma tela utilizada para desenhos semelhante às utilizadas no AutoCAD, visto na figura 2.

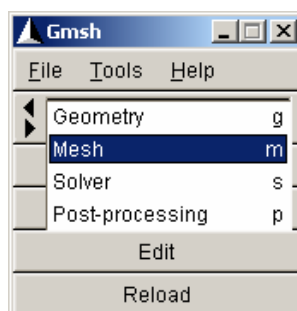


Figura 1. Visualização dos módulos do GmsH



Figura 2. Tela de CAD

1.1. Como criar uma malha:

Para se criar qualquer tipo de malha, deve-se primeiro criar a geometria respectiva a superfície que se deseja malhar. A criação de uma geometria deve seguir os seguintes passos: Clica-se em Elementary, Add, New para criar “entidades” geométricas elementares (ponto, reta, arco de circunferência, splines, etc.), como na figura 3 abaixo.

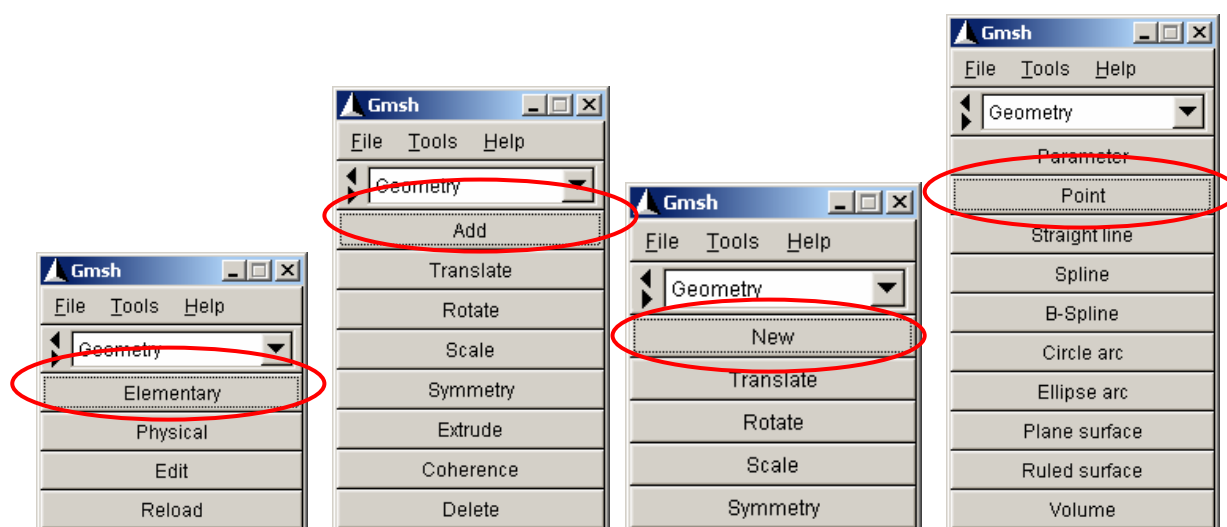


Fig. 3 – Sequência de comandos para criar uma geometria

Ao se clicar em *Point*, visualizar-se-á uma janela onde serão inseridas as coordenadas dos pontos bem como seu comprimento. As coordenadas podem ser inseridas manualmente na tela de CAD, posicionando o mouse na coordenada desejada seguido da tecla “e”.

Abaixo segue um exemplo para o ponto com coordenadas $(x,y,z) = (4,0,0)$ e comprimento 0.1, mostrado na figura 4.

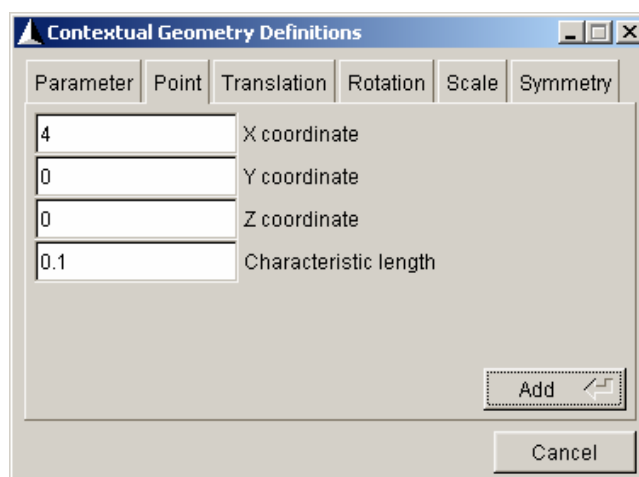


Fig. 4 - coordenadas dos pontos

Quando se cria uma seqüência de pontos, um arquivo de extensão “.geo”, figura 5, é criado e pode ser visualizado no módulo “Geometry” com a opção “Edit”. Sendo assim podemos criar geometrias, figura 6, sem utilizar interface gráfica, apenas editando um documento texto.

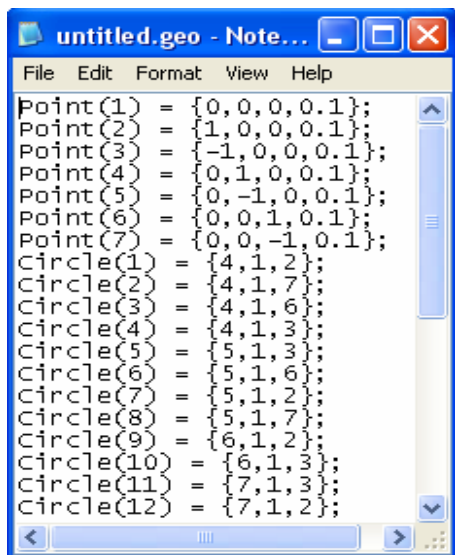


Fig. 5 – arquivo “.geo” da geometria

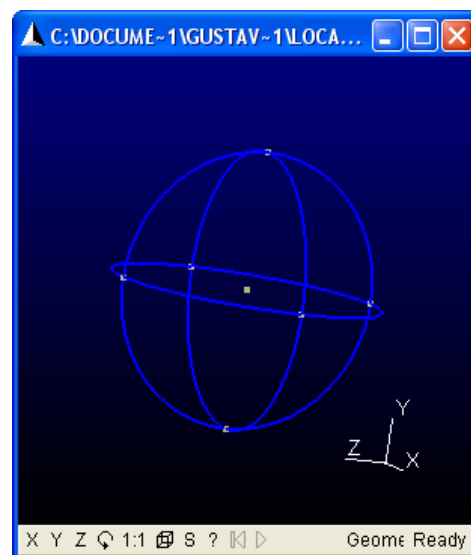


Fig. 6 – visualização da geometria

No caso de geometrias tridimensionais deve-se levar em consideração o eixo dos planos cartesianos, que disponibiliza uma referência da orientação da figura criada, como mostrado na figura 7.

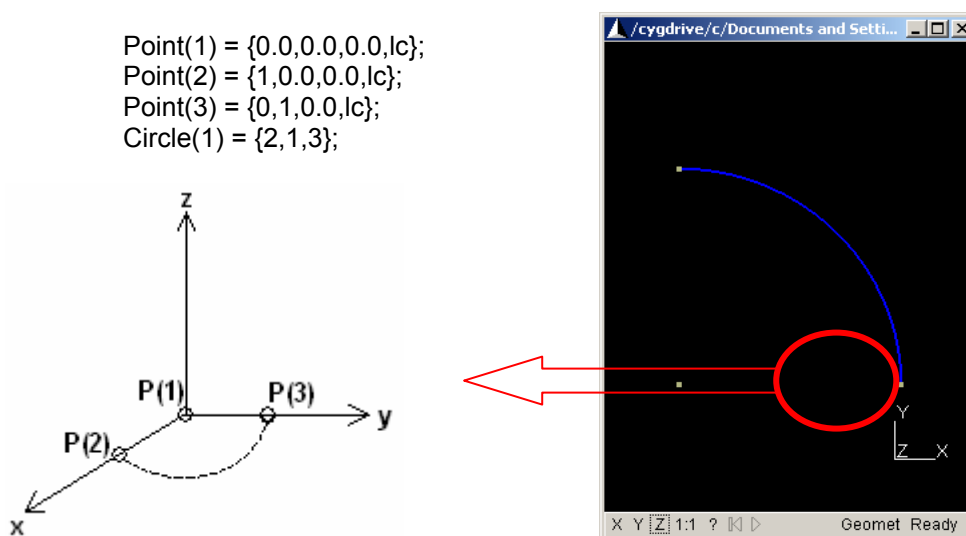


Fig. 7 – Referência com o eixo das coordenadas

Para indicar os limites de uma superfície seleciona-se a opção *Ruled surface* ou

Plane Surface, dependendo de o formato do plano ser curvo ou não. Ao clicar em alguma linha, o gmsh tenta construir a superfície. Para confirmar a criação da superfície, tecla-se “e”.

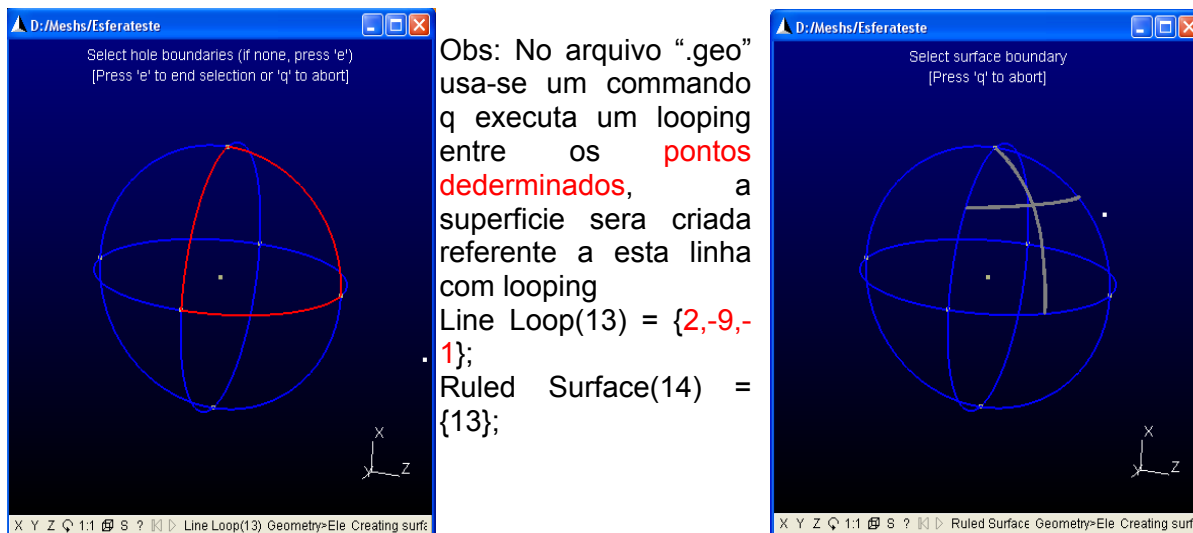


Fig. 8 – Determinando uma superfície na geometria

Para criar a malha, deve-se selecionar o módulo Mesh, como não determinamos nenhum volume, vamos criar uma malha somente sobre a superfície:

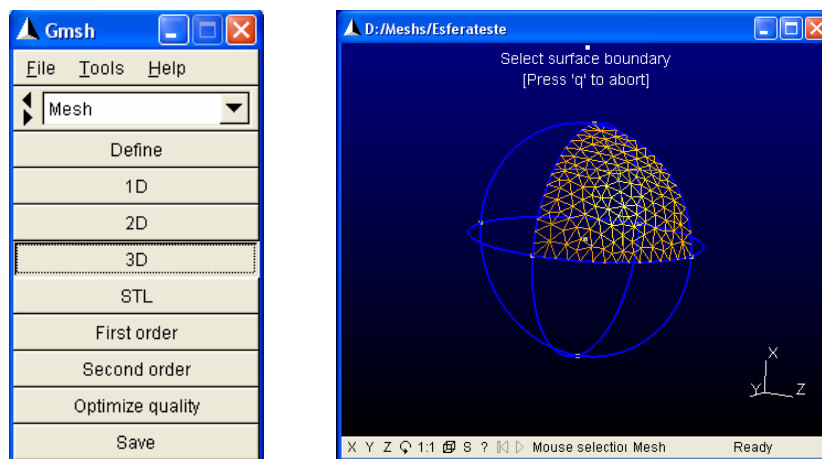


Fig. 9 – Malha sobre uma superfície

Após determinar todas as superfícies que se deseja na geometria, pode-se gerar a malha completa da geometria.

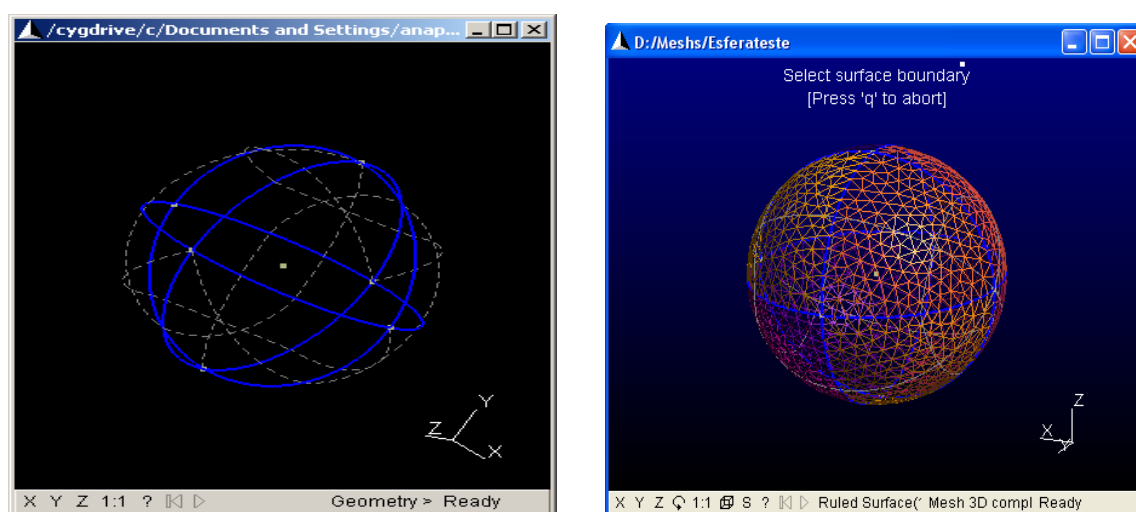


Fig. 10 – Malha completa após determinação das superfícies

1.2. Como refinar uma malha:

De acordo com a finalidade para qual serão usadas as geometrias e as malhas, existe a possibilidade de se aperfeiçoar a qualidade das mesmas. A malha triangular padrão do Gmsh, vista na figura 10, é gerada com elementos o mais uniforme possível, podendo haver algumas deformações.

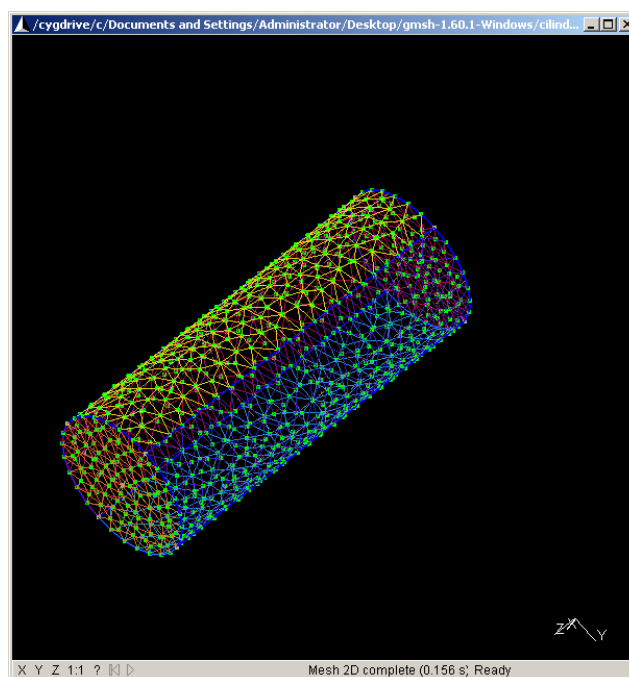


Fig. 11 – Malha triangular padrão do GMSH

A possibilidade de refinar a malha das geometrias é viável principalmente quando

essa geometria é formada por composição de várias faces, de forma a gerar geometrias complexas como automóveis e aviões. Tomando como exemplo a última geometria citada, podemos analisar que, pelo fato do avião possuir vários encontros de superfícies, e possibilidade de deformação da malha é muito grande, Figura 12.

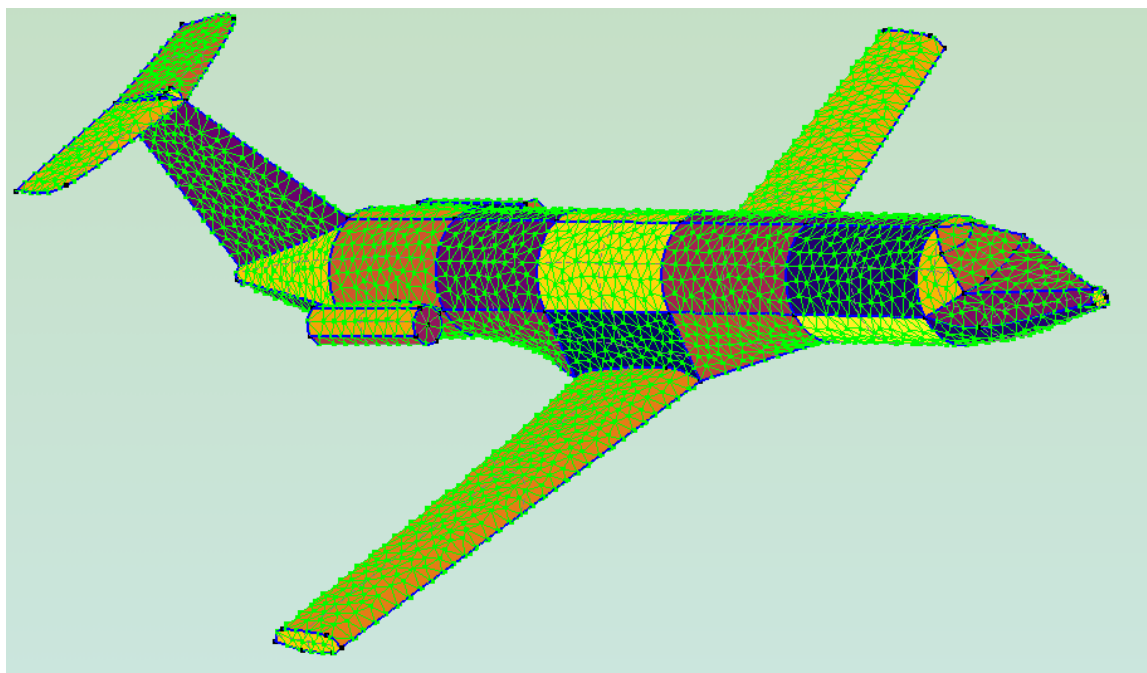


Figura 12. Detalhes de Geometria de avião composta por várias superfícies.

Para o arquivo de malhas, a necessidade de refinamento não é de grande importância, uma vez que o arquivo mais usado para simulação computacional em CFD é o de formato .STL. Sendo assim, a maior preocupação de refinamento de malha é na geração deste tipo de arquivo, onde se torna visível às deformações da malha, figura 13.

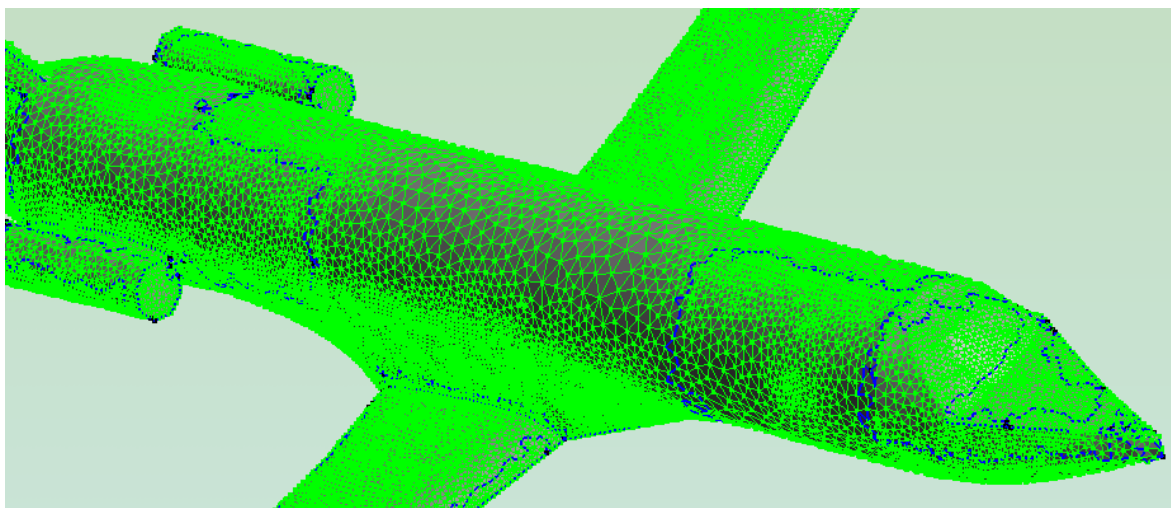


Figura 13. Detalhes do arquivo .STL com deformação da malha superficial.

Portanto, nestes casos faz-se necessário refinar a malha computacional para gerar o respectivo arquivo em STL. Este refinamento é feito no GMSH, através da configuração de alguns parâmetros.

Para alterar estes parâmetros, basta através da interface do GMSH, acionar na barra de ferramentas a opção “Tools” e depois “Options”, assim a caixa de configuração para essas informações se abrirá, como mostrado na figura 14 abaixo.

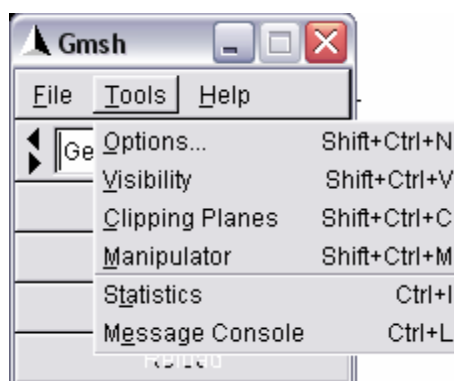


Figura 14. Chamando a área para configuração do refinamento.

Para refinamento, tem-se a opção de refinar a malha e o STL que compões a superfície. Na figura 15 abaixo, visualiza-se a área de configuração do refinamento da malha que pode ser alterada mudando o valor “Characteristic length factor”.

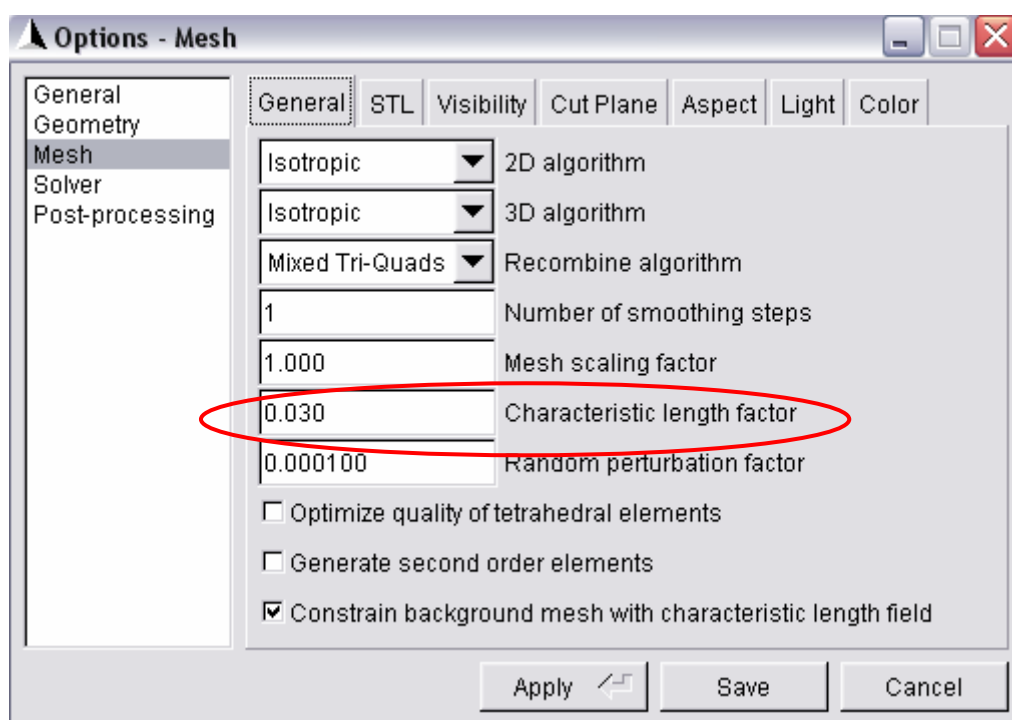


Figura 15. Área de configuração do tamanho da malha.

Porém para o refinamento do arquivo utilizado para simulações computacionais, o real interesse é modificar a estrutura da casca da geometria, através de alterações dos parâmetros abaixo selecionados. Nota-se que eles dependem do parâmetro modificado anteriormente na malha, pois se efetua uma divisão do fator de tamanho característico configurado para malha, pelos parâmetros de tamanho de elemento mínimo e tamanho de elemento objetivado, figura 16.

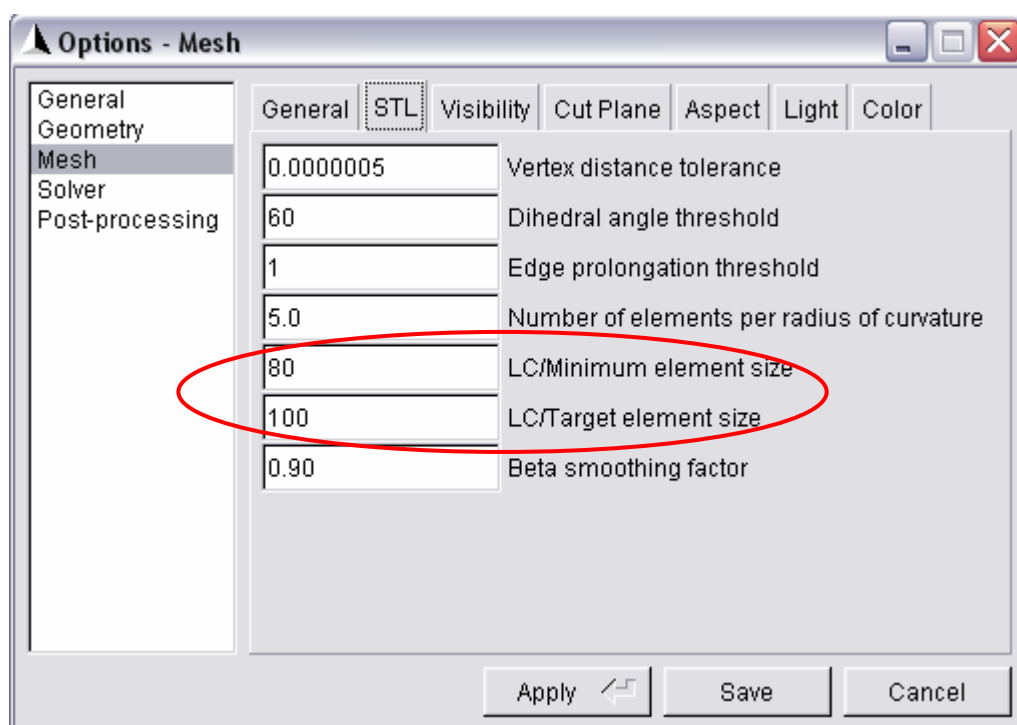


Figura 16. Área de configuração do refinamento no arquivo STL.

Sendo assim, configurando-se estes parâmetros, consegue-se obter uma malha homogênea a contento.

1.3. Bibliografia

GEUZAIN, C., et al. "Gmsh Reference Manual: The documentation for Gmsh 1.60", 2005. Disponível em: <<http://www.geuz.org/gmsh/>> Acessado dia: 20/08/2005;

Anexo 2. Desenvolvimento de Cluster de PCs Para Solução de Problemas de Mecânica de Fluidos Computacional

2.1. Introdução

O objetivo geral deste tutorial é o desenvolvimento de um cluster (conjunto de PCs interligados via rede) do tipo Beowulf para o Laboratório de Transferência de Calor e Massa e Dinâmica de Fluidos (LTCM), que resolva problemas contendo uma grande massa de dados.

O LTCM foi criado visando o desenvolvimento de modelos matemáticos, métodos numéricos e técnicas experimentais que possibilitem o estudo da transferência de calor e massa e da dinâmica dos fluidos. Neste contexto, o projeto de Iniciação Científica desenvolvido tem o papel de fornecer base computacional de grande porte e com grande poder de processamento a fim de simular tais modelos.

Dentre os problemas que são abordados no LTCM, os que exigem maior demanda de processamento são os relativos à dinâmica de fluidos, visto a complexidade das equações governantes.

O projeto abrange especificação das máquinas, configuração, manutenção e gerenciamento do cluster, estudo, implementação e análise de programas paralelos utilizando MPI e, finalmente, possibilitar a aplicação de técnicas paralelas em modelos matemáticos complexos como na equação de Navier-Stokes.

O rápido aumento na capacidade de um único processador tem diminuído nos últimos anos. Atualmente, para se obter um grande aumento em velocidade de processamento é necessária a utilização de computadores multiprocessados ou paralelos. A vantagem destes computadores em relação aos supercomputadores que utilizam técnicas de vetorização, é a escalabilidade e o baixo custo. Computadores paralelos disponíveis comercialmente podem ter milhares de processadores, gigabytes de memória e poder computacional medido em gigaflops. No entanto, algoritmos projetados para máquinas seriais podem não ser tão eficientes em máquinas paralelas.

Uma opção para a implementação de um sistema paralelo seria a utilização de máquinas com mais de um processador utilizando memória compartilhada, ou seja, um PC

com dois ou mais processadores utilizando recursos compartilhados como: memória, disco rígido e periférico. No entanto, esta implementação torna-se inviável quando se deseja escalabilidade no sistema. O número de processadores que pode ser adicionado em uma máquina é limitado.

A melhor opção, considerando as necessidades do laboratório, é a implementação de um cluster de PCs, mais especificamente um cluster Beowulf.

A tecnologia de cluster começou nas máquinas de alto desempenho (supercomputadores). O desenvolvimento dessa tecnologia em PC's foi motivado devido ao alto preço dos supercomputadores, começando em 1994 com o desenvolvimento do cluster Beowulf pela NASA.

2.2. Cluster Beowulf

Em 1994, a NASA, agência espacial norte-americana, necessitava de um equipamento com poder de processamento da ordem de um gigaflop. Uma máquina de tecnologia RISC com tal nível de desempenho custava em torno de um milhão de dólares. Diante disso, os pesquisadores Thomas Sterling e Donald J. Becker decidiram então interligar 16 computadores pessoais, cada um com um microprocessador 486, usando Linux e uma rede ethernet. Este conjunto atingiu a marca de 70 megaflops. Para a época, era uma velocidade comparável a de pequenos supercomputadores comerciais. O custo total do sistema foi de quarenta mil dólares ou seja, dez por cento do preço de uma máquina equivalente para a época.

Uma característica chave de um cluster Beowulf é o uso do Linux e de bibliotecas para troca de mensagens de livre distribuição, permitindo assim alterações no sistema operacional.

Um cluster pode possuir vários tipos de configurações diferentes, tanto na montagem do hardware quanto na configuração do sistema. Os tipos mais comuns de Clusters linux são:

Cluster de processamento paralelo: cada vez que o cluster recebe uma tarefa para executar, já previamente preparada para rodar em processamento paralelo, o cluster divide os pedaços da tarefa para cada uma das máquinas realizar. Dessa forma, com várias máquinas trabalhando ao mesmo tempo em uma única tarefa, o tempo necessário para executá-la torna-se consideravelmente menor.

Cluster de disponibilidade: esse tipo de cluster funciona como um gerenciador de tarefas, ou seja, cada máquina trabalha sozinha, porém a distribuição de tarefas é feita de

tal forma que os processadores estão sempre trabalhando na capacidade total. Nesse tipo de cluster é vital a implementação de um sistema de filas com vários níveis de prioridades diferentes, onde o servidor de filas irá gerenciar qual processador ficará com qual tarefa e o quanto de sua capacidade será utilizado para cada tarefa. Esse tipo de cluster é ideal para trabalhar com grandes quantidades de tarefas que exigem pequenas ou médias capacidades de processamento.

Dentre os tipos de Clusters Linux citados, o que mais se encaixa aos propósitos do LTCM é o Cluster de processamento paralelo, visto a necessidade de dividir tarefas entre vários computadores. Esta necessidade é justificada devido ao grande volume de dados que deverão ser alocados e processados pelo sistema.

2.3. Os passos rumo à paralelização

O primeiro passo consiste em buscar uma plataforma que torne viável o uso de ambientes distribuídos e utilizando apenas produtos de distribuição gratuita. A plataforma escolhida é o Linux, que oferece uma vasta quantidade recursos, e a distribuição inicialmente utilizada foi RedHat 7.1. Foi escolhido, também, a biblioteca MPI para a troca de mensagens entre os processos disponíveis no cluster. Para iniciar os testes foi adquirido 3 computadores idênticos ligados a uma rede Ethernet trabalhando a uma taxa de 100Mbps/s.

Antes de iniciar a caminhada rumo à paralelização de programas voltados ao Laboratório, foram realizados testes usando programas paralelos distribuídos gratuitamente na Internet. Estes testes iniciais foram importantes para se verificar a eficiência do sistema. Comprovado o ganho em desempenho de programas paralelos, passou-se a etapa de implementação de programas que estão relacionados à natureza dos problemas do LTCM. Foi implementado um programa paralelo, usando MPI, que avalia a condução de calor em um domínio tridimensional. Em virtude dos resultados, seguiu-se para uma nova etapa: a implementação do primeiro programa paralelo do LTCM para solução das equações de Navier-Stokes (já estando em fase avançada de testes). Essas equações descrevem com bastante precisão o comportamento do escoamento de fluidos.

Para esta etapa do projeto, o LTCM conta com novas máquinas que são capazes de simular problemas que ultrapassem os 10 milhões de nós. Este trabalho vai detalhar passo a passo a construção deste cluster, desde suas especificações à implementação dos programas paralelos.

2.4. Especificações

2.4.1. Hardware

Um cluster Beowulf pode ser implementado usando máquinas homogêneas ou heterogêneas. Um cluster é dito homogêneo quando todos os seus nós possuem as mesmas características e a mesma rede de comunicação interligando-os. Um cluster heterogêneo possui nós que apresentam características diferentes ou diferentes redes de comunicação entre grupos de máquinas. A implicação é que, em máquinas homogêneas, realizar o balanceamento de carga torna-se mais fácil, visto que todos os nós apresentam as mesmas características. Já em cluster heterogêneo, uma análise mais profunda deve ser feita levando-se em consideração as diferentes capacidades de processamento de cada nó.

Atualmente, o LTCM conta com 5 computadores, um “front-end” (mestre) e quatro escravos com características que o constitui um cluster homogêneo. As configurações são listadas logo abaixo.

Configurações de hardware do mestre:

- Placa-mãe: Intel 865PERL;
- Placa de rede: e1000 1000MB/s on-board;
- Placa de rede: 3Com 100MB/s;
- Processador: Intel Pentium 4 2.8GHz;
- Memória: 1024 MB DDR;
- HD: 80GB IDE;
- CDR-W: HP cd-writer 9300 series;
- Placa de vídeo: Radeon 9200 128 MB DDR AGP8x.
- Monitor: Samsung SincMaster 753v de 17";
- Periféricos: teclado e mouse ótico.

Configurações de hardware dos escravos:

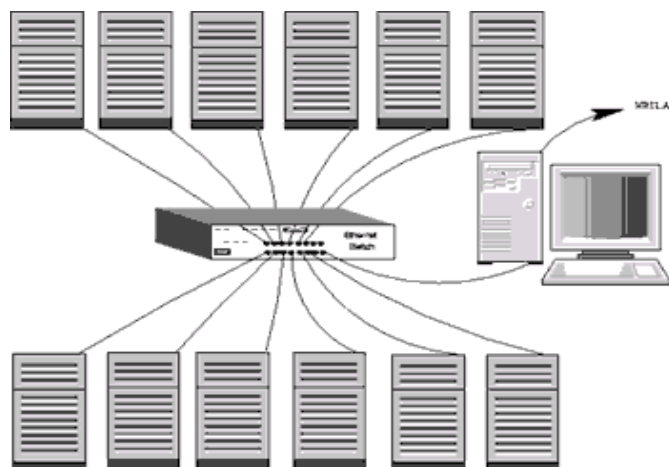
- Placa-mãe: Intel 865PERL;
- Placa de rede: e1000 1000MB/s on-board;
- Processador: Intel Pentium 4 2.8GHz;
- Memória: 1024 MB DDR;
- HD: 80GB IDE;

- CDROM: LG 52X;
- Placa de vídeo: Geforce 64 MB AGP4x.

Estrutura da Rede Local

- Switch: 3COM 8port Gigabit modelo 3C16477
- Cabos: 5 cabos par trançado de 1Gigabit

A arquitetura do cluster pode ser ilustrada na figura abaixo.



Como já foi dito, a escolha das configurações deve-se à enorme demanda de memória e processamento de dados, além da alta comunicação necessária entre os nodos do cluster.

2.4.2. Software

O sistema operacional utilizado é o Linux por ser público, gratuito, e possuir código fonte aberto. A distribuição utilizada é o RedHat 9.0.

O computador principal (front-end) é o equipamento na qual está instalado:

- NFS;
- Serviços de acesso remoto: rlogin, ssh, ftp, rsh, telnet;
- Bibliotecas paralelas: MPICH 1.2.5;

- Compilador Fortran da Intel para Linux;
- Aplicação de gerência de nós: bWatch;
- Analisadores de Performance: NetPipe, NetPerf, Vampir, VampirTrace;
- Aplicativos de visualização: Paraview, Opendx;

Explicitando um pouco mais, o computador mestre funciona como servidor de arquivos para os computadores escravos, utilizando o protocolo NFS, e também como porta de entrada para os usuários que desejem utilizá-lo.

A tarefa dos computadores escravos é a execução exclusiva das tarefas paralelizadas. Em geral, não é necessária a utilização de monitor, mouse e teclado nos computadores escravos, sendo todos eles gerenciados e controlados remotamente pelo computador mestre através de serviços como ssh. Como os computadores escravos não são acessados diretamente, foi reservado o endereçamento IP 192.168.1.1/5 para este fim.

A coordenação das tarefas é feita por troca de mensagens de um computador para outro. Há duas bibliotecas populares de troca de mensagens: a Interface de Passagem de Mensagens (MPI – Message Passing Interface) e a Máquina Virtual Paralela (PVM – Parallel Virtual Machine), ambas disponíveis gratuitamente por seus mantenedores via Internet.

Como a maioria dos programas desenvolvidos no laboratório são em Fortran 90, foi escolhido o compilador Fortran da Intel. Este compilador, além possuir licença gratuita apresenta várias opções de otimização.

Para monitoração foi utilizada a ferramenta bWatch. Sua tarefa é observar a carga e o uso da memória em todos os nós do cluster. Ferramentas de análise como NetPipe também foi instalada para avaliar a performance da rede.

Para a visualização gráfica dos resultados 3D, está sendo utilizado o Paraview e o Opendx, que apresentam características de processamento paralelo.

2.5. Instalação e configuração do cluster

Este tópico descreve, de forma resumida, os passos seguidos para a implantação do cluster. Será abordadas a instalação do Linux, configuração da rede local, configuração de alguns serviços e instalação de pacotes adicionais.

2.5.1. Instalando o Linux

O primeiro passo na montagem do cluster é a instalação do Linux em todas as máquinas. Durante esse processo, é interessante verificar se todos os pacotes necessários estão incluídos, principalmente o NFS e os serviços de acesso remoto (rlogin, ssh, ftp, rsh, telnet). A distribuição Linux escolhida para a implementação do cluster no LTCM é Red Hat 9.

Instalado o Linux em todos os nós, o próximo passo é a configuração da rede, que também pode ser feito durante a instalação, caso tenha sido detectada a placa de rede.

2.5.2. Configurando serviços

2.5.2.1. RSH

A configuração da rede local está definida da seguinte forma:

IP da rede local: 192.168.1.0

Máscara de subrede: 255.255.255.0

Nome do servidor: bwf01

Nomes dos nós escravos: bwf02, bwf03, bwf04 e bwf05.

Os IPs de cada máquina foram definidos da seguinte forma:

bwf01 192.168.1.1

bwf02 192.168.1.2

bwf03 192.168.1.3

bwf04 192.168.1.4

bwf05 192.168.1.5

Em todos os computadores foi adicionado as seguintes linhas ao arquivo `/etc/hosts/`:

127.0.0.1 localhost localhost.localdomain

192.168.1.1 bwf01

192.168.1.2 bwf02

192.168.1.3 bwf03

```
192.168.1.4   bwf04
192.168.1.5   bwf05
```

Este arquivo contém as informações de todos os computadores que estão conectados à rede local.

O próximo passo será configurar o arquivo `/etc/hosts.equiv`, que define a relação de confiança entre os hosts através de equivalência, sem a necessidade de autenticação por senha. Isto representa um grande risco de segurança, mas é requerido pelo protocolo de acesso remoto RSH (Remote Shell) para que seja possível acessar todas as máquinas do cluster. Este arquivo deve estar presente em todas as máquinas que farão parte do seu sistema. O formato é o seguinte:

```
bwf01
bwf02
bwf03
bwf04
bwf05
```

Outro arquivo que deverá ser editado é o arquivo `.rhosts`. Este arquivo deverá constar em cada diretório de trabalho de cada usuário, como `/home` e `/root`. O ponto (.) na frente do arquivo o torna “invisível” ao comando `ls`. Este arquivo também será usado pelo protocolo RSH para execução de comandos remotos, por algumas aplicações de monitoramento. Seu conteúdo é o seguinte:

```
bwf01
bwf02
bwf03
bwf04
bwf05
```

É importante salientar que os arquivos `hosts`, `hosts.equiv` e `.rhosts` devem ser replicados em todas as máquinas que formam o cluster.

Habilite os serviços de `rsh`, `rlogin`, `ssh` e `rexec` com o comando `ntsysv`.

Finalmente, acrescente as seguintes linhas ao final do arquivo `/etc/securetty/`:

```
rsh
rlogin
```

2.5.2.2. Instalando e configurando o compilador Fortran da Intel

O compilador Fortran da Intel pode ser baixado, junto com a licença, no *site* <http://www.intel.com/software/products/compilers/flin/>. Feito o download, basta descompactar e incluir no diretório criado a licença gratuita. A descrição deste tópico refere-se à instalação e configuração do compilador Fortran versão 7.x.

A instalação do pacote é bem simples, basta executar um script de instalação dentro do diretório que foi criado digitando:

```
sh install.sh
```

Erros na instalação podem ocorrer devido a falta da licença de uso.

O próximo passo é compilar o arquivo `ifc` (que está localizado em `/opt/intel/compiler70/ia32/bin/` no caso da instalação padrão) em `/usr/bin`. Isso será necessário posteriormente para que a configuração do `mpich` verifique a existência do compilador, permitindo que os códigos em Fortran paralelos sejam compilados usando o `ifc`.

Um problema verificado foi a incompatibilidade entre o compilador Intel `ifc` versão 7.1 e o Red Hat 9.0 ou outra versão Linux com o mesmo `clib` (isto está também documentado no *site* da Intel). Para resolver o problema basta criar um arquivo `aaa.c` em `/opt/intel/compiler70/ia32/lib` contendo as seguintes linhas:

```
int errno=0;
const unsigned short int __ctype_b = 0;
```

Em seguida o arquivo é compilado da seguinte forma:

```
gcc -O3 -c aaa.c
```

Finalmente edit o arquivo `ifc.cfg` em `/opt/intel/compiler70/ia32/bin` da seguinte forma:

```
/opt/intel/compiler70/ia32/lib/aaa.o      -Xlinker      -rpath      -Xlinker
/opt/intel/compiler70/ia32/lib -l/opt/intel/compiler70/ia32/include
```

2.5.2.3. Instalando o MPICH

MPICH é uma implementação portátil do MPI (biblioteca de passagem de mensagens desenvolvida para ser padrão de comunicação entre processos em ambientes de memória distribuída) distribuída livremente. A versão gratuita desta biblioteca pode ser obtida no *site* <http://www-unix.mcs.anl.gov/mpi/mpich/>.

O programa baixado pode ser descompactado no diretório `/usr/local` da seguinte forma:

```
tar -zxvf mpich.tar.gz
```

Agora basta entrar no diretório `mpich-1.2.5` recém-criado e executar a seqüência de comandos abaixo:

```
./configure -fc=ifc -f90=ifc --prefix=/usr/local/mpich-1.2.5  
make  
make install
```

As opções `-fc` e `-f90` permitem que a configuração do `mpich` inclua a utilização do compilador `ifc` previamente instalado.

Agora basta editar o arquivo `machines.LINUX`, localizado no diretório `/usr/local/mpich-1.2.5/util/machines/`, e acrescentar os nomes dos computadores que farão parte do cluster.

```
bwf01  
bwf02  
bwf03  
bwf04  
bwf05
```

Caso se deseje utilizar SSH no lugar de RSH, basta configurar o `mpich` da seguinte forma:

```
./configure -rsh=ssh -fc=ifc -f90=ifc --prefix=/usr/local/mpich-1.2.5  
make  
make install
```

2.5.2.4. Configurando o Servidor do Sistema de Arquivos (NFS)

O servidor NFS tem a função de permitir o acesso transparente a discos remotos, centralizando assim a administração, pois é possível ter todos os diretórios de trabalho em uma única máquina e compartilhá-lo com diversos clientes conectados por uma rede local.

Para o cluster do LTCM será necessário o compartilhamento de dois diretórios. Um contém a instalação da versão gratuita da biblioteca MPI (mpich) e o outro a instalação do compilador Fortran da Intel. A utilização do NFS torna desnecessária a instalação destes pacotes em outras máquinas.

Para isso, no computador mestre, o arquivo `/etc/exports` foi editado acrescentando as seguintes linhas:

```
/usr/local/mpich-1.2.5 *(rw,no_root_squash,sync)
/opt/intel          *(rw,no_root_squash,sync)
```

Com arquivo salvo basta iniciar o serviço NFS digitando o seguinte comando:

```
/etc/init.d/nfs start
```

Para configurar os clientes, em cada nó escravo deve ser editado o arquivo `/etc/fstab` para montar automaticamente o sistema de arquivos remoto. Para tanto, basta adicionar as seguintes linhas ao final do arquivo:

```
192.168.1.1:/usr/local/mpich-1.2.5 /usr/local/mpich-1.2.5 nfs exec,dev,suid,rw 1 1
192.168.1.1:/opt/intel /opt/intel nfs exec,dev,suid,rw 1 1
```

Não esquecendo que devem ser criados os diretórios `/usr/local/mpich-1.2.5` e `/opt/intel`, onde serão os pontos de montagem do sistema de arquivos remoto.

E, finalmente, para verificar se o cluster está funcionando basta entrar no diretório `/usr/local/mpich-1.2.5/util/` e digitar:

```
cd /usr/local/mpich-1.2.5/util/
./tstmachines -v
```

Uma mensagem como a ilustrada abaixo, indica a correta configuração e o bom funcionamento do cluster.

Trying true on bwf01 ...
Trying true on bwf02 ...
Trying true on bwf03 ...
Trying true on bwf04 ...
Trying true on bwf05 ...
Trying ls on bwf01 ...
Trying ls on bwf02 ...
Trying ls on bwf03 ...
Trying ls on bwf04 ...
Trying ls on bwf05 ...
Trying user program on bwf01 ...
Trying user program on bwf02 ...
Trying user program on bwf03 ...
Trying user program on bwf04 ...
Trying user program on bwf05 ...

2.6. Desenvolvimento de programas paralelos

Desenvolver programas paralelos baseados em bibliotecas *Message-Passing* não é nada trivial. Por isso, antes de começar a trabalhar com programas paralelos, é preciso entender alguns conceitos fundamentais de processamento paralelo.

2.6.1. Paralelismo

Paralelismo é uma técnica usada para dividir tarefas grandes e complexas em tarefas pequenas obtendo, dessa forma, resultados mais rápidos. Essas tarefas podem ser distribuídas em vários processadores para serem executadas simultaneamente.

As principais vantagens dessa técnica são:

- ♦ Aumento do desempenho no processamento;
- ♦ Solução de problemas computacionais grandes e complexos.

2.6.2. Ambiente paralelo

O ambiente paralelo consiste em vários processadores interligados em rede, constituído por uma plataforma para manipulação de processos paralelos. Uma das formas de estabelecer a comunicação entre os processadores é por troca de Mensagens (Message Passing). Este é o método de comunicação baseada no envio e recebimento de mensagens através da rede seguindo as regras do protocolo de comunicação entre vários processadores que possuam memória própria. O programador é responsável pela sincronização das tarefas. Como exemplo temos:

PVM - Parallel Virtual Machine

MPI - Message Passing Interface

MPL - Message Passing Library

2.6.3. Programação Paralela

O objetivo da programação paralela é dividir algoritmos grandes e complexos em pequenas tarefas executadas simultaneamente por vários processadores que se comunicam entre si.

De forma bem simples, os passos para se desenvolver um programa paralelo podem ser resumidos da seguinte forma:

- ◆ Decompor o algoritmo ou os dados;
- ◆ Distribuir as tarefas ou dados entre vários processadores que trabalham simultaneamente;
- ◆ Coordenar os processos em execução e a comunicação entre os processadores (sincronização).

2.6.4. Obstáculos do Paralelismo

2.6.4.1. Sincronização entre os processos

As tarefas executadas em paralelo, num determinado instante, aguardam a finalização mútua para coordenar os resultados ou trocar dados e reiniciar novas tarefas em paralelo.

É necessário que haja a coordenação dos processos e da comunicação entre eles

para evitar que a comunicação seja maior do que o processamento e que consequentemente haja uma queda no desempenho dos processos em execução.

2.6.4.2. Compiladores e ferramentas em amadurecimento

Existem poucos compiladores e ferramentas prontas para paralelização automática que resolvam definitivamente o problema do paralelismo.

2.6.4.3. A conversão de algoritmos seqüenciais em paralelos

Há um considerável tempo gasto do programador em analisar o código fonte para paralelizar e recodificar.

Basicamente é necessário rever o programa seqüencial, avaliar como será particionado, quais os pontos de sincronização, quais as partições que poderão ser executadas em paralelo e qual a forma de comunicação que será empregada entre as tarefas paralelas.

2.6.4.4. Nem tudo é paralelizável

É necessário que haja uma análise do algoritmo a ser paralelizado para seja possível a paralelização nos dados ou nas funções do mesmo, levando sempre em consideração a quantidade de comunicação em relação ao processamento.

2.6.4.5. Tempo de processamento

No processamento paralelo, o tempo de CPU quase sempre aumenta, no entanto pode-se reduzir o tempo total de processamento.

2.6.4.6. Tecnologia Nova

Por ser uma tecnologia recente, há uma dificuldade em se desenvolver aplicações

usando paralelismo.

2.6.4.7. Perda de Portabilidade

A programação paralela utiliza multiprocessadores com arquiteturas de comunicação entre os processadores baseados em memória compartilhada (shared memory) e memória distribuída (distributed memory).

Os programas adaptados a arquitetura de comunicação entre os processadores baseado em memória compartilhada não podem ser executados em uma máquina com memória distribuída, pois não utilizam os mecanismos de troca de mensagens, impossibilitando assim a portabilidade.

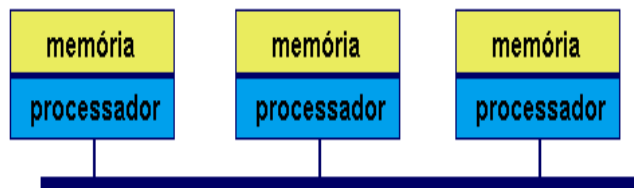
No caso inverso, memória distribuída para memória compartilhada, seria possível a portabilidade; porém o mecanismo de troca de mensagem seria ignorado, pois todos os dados já poderiam ser acessados pelos processadores através da memória compartilhada, isto apenas traria um fluxo desnecessário de comunicação.

2.6.4.8. Depuração (*Debug*)

Os processos são distribuídos e executados em vários processadores simultaneamente, entretanto não existe uma forma eficiente de acompanhar passo-a-passo a alteração das variáveis durante a execução do processamento das diversas tarefas paralelas.

2.6.4.9. Modelo de memória distribuída

Num sistema distribuído os processadores operam independentemente, onde cada um possui sua própria memória. Os dados são compartilhados através da rede usando um mecanismo de troca de mensagens (Message Passing).



2.7. Lei de Amdahl

Lei de Amdahl determina o potencial de aumento de velocidade a partir da porcentagem paralelizável do programa.

$$Speedup = \frac{1}{1 - \% \text{ paralelizável}}$$

Consideremos uma aplicação que leva T unidades de tempo quando executado em modo serial em um único processador.

Quando a aplicação é paralelizável, assumimos que o parâmetro S constitui o percentual serial e P o percentual que pode ser paralelizado.

Assim temos:

Tempo Sequencial:

$$T = T_s$$

Tempo Paralelo:

Assumindo N processadores em implementação paralela, o tempo de execução é dado pela seguinte expressão (assumindo speedup perfeito na parte da aplicação que pode ser paralelizada).

$$T_p = \left(S + \frac{P}{N} \right) \cdot T$$

O speedup que é obtido por N processadores é:

$$S_p = \frac{T_s}{T_p} = \frac{1}{S + \frac{P}{N}}$$

Ou seja:

$$\text{Speedup} = \frac{1}{\frac{P}{N} + S}$$

Onde:

P = % paralelizável

N = N° de processadores

S = % serial

Assume-se que a parte serial é fixa, então o speedup para infinitos processadores é limitado em:

$$\text{Speedup} = 1/S$$

Por exemplo, se Speedup = 10%, então o speedup máximo é 10, até se usarmos um número infinito de processadores.

Tabela Comparativa: Para compreender melhor o significado da Lei de Ahmdahl, veja como se comportariam os casos hipotéticos da tabela abaixo:

N	P=0,50	P=0,90	P=0,99
10	1,82	5,26	9,17
100	1,98	9,17	50,25
1.000	1,99	9,91	90,99
10.000	1,99	9,91	99,02

2.8. Granulosidade

A granulosidade de um aplicação paralela é uma medida usada para indicar a relação entre o tamanho de cada tarefa e o tamanho total do programa, ou seja, é a razão entre computação e comunicação.

2.8.1. Granulosidade fina

Isto ocorre quando um problema é dividido em tarefas muito pequenas, ou seja, os cálculos executados em cada processador são mínimos. Uma implementação paralela com esta característica facilita o balanceamento de carga em contrapartida, há uma maior frequência de comunicação, a performance do programa tende a diminuir e o custo de sincronização é alto.

2.8.2. Granulosidade grossa

Neste caso o problema é dividido em tarefas que sejam maiores possíveis. Isto tem como características: aumentar a performance do programa, processamento menor que a comunicação, menor custo de sincronização, no entanto dificulta o balanceamento de carga.

Para se determinar qual a melhor granulosidade deve-se analisar o grau de dependência entre os processos, definir o grau de sincronização necessário. Em alguns casos a própria natureza do problema pode limitar a granulosidade possível.

2.9. Balanceamento de carga

O balanceamento de carga é um fator importante a ser analisado quando se deseja implementar um programa paralelo. Refere-se à distribuição otimizada dos processos entre os processadores fazendo com que o tempo de execução dos processos em paralelo seja eficiente.

2.10. Escalabilidade

É a propriedade de um sistema paralelo de aumentar o speedup a medida que aumentamos o número de processadores.

Obstáculos:

- ◆ *Hardware*

Um dos maiores gargalos é o tráfego de informação na rede que interliga os processadores. O aumento do número de processadores pode degradar a capacidade de comunicação da rede, diminuindo a performance do programa.

- ◆ *Software*

Alguns algoritmos trabalham melhor em certas escalas de paralelismo. O aumento do número de processadores pode acarretar uma queda na eficiência durante a execução do programa.

2.11. Modelo Message Passing

O modelo de Message Passing é um conjunto de processos que possuem acesso à memória local. As informações são enviadas da memória local do processo para a memória local do processo remoto.

A comunicação dos processos é baseada no envio e recebimento de mensagens.

A transferência dos dados entre os processos requer operações de cooperação entre cada processo de forma que operação de envio deve casar com uma operação de recebimento.

O modelo computacional Message Passing não inclui sintaxe de linguagem nem biblioteca e é completamente independente do hardware. O paradigma de passagem de mensagem apresenta-se apenas como uma das alternativas mais viáveis, devido a muitos pontos fortes como:

- ◆ Generalidade: Pode-se construir um mecanismo de passagem de mensagens para qualquer linguagem, como ferramentas de extensão das mesmas (bibliotecas);
- ◆ Adequação à ambientes distribuídos.

Porém, apresenta limitações:

- ◆ O programador é diretamente responsável pela paralelização;

- ♦ Custos de comunicação podem tornar extremamente proibitiva a transmissão de mensagens em um dado ambiente.

2.12. MPI

MPI é uma biblioteca de Message Passing desenvolvida para ambientes de memória distribuída, máquinas paralelas massivas, NOWs (network of workstations) e redes heterogêneas. Define um conjunto de rotinas para facilitar a comunicação (troca de dados e sincronização) entre processos paralelos.

A biblioteca MPI é portátil para qualquer arquitetura, tem aproximadamente 125 funções para programação e ferramentas para se analisar a performance.

A biblioteca MPI possui rotinas para programas em Fortran 77 e ANSI C, portanto pode ser usada também para Fortran 90 e C++. Os programas são compilados e linkados a biblioteca MPI.

Todo paralelismo é explícito, ou seja, o programador é responsável por identificar o paralelismo e implementar o algoritmo utilizando chamadas aos comandos da biblioteca MPI.

2.12.1. Conceitos básicos de MPI

Processo:

Por definição, cada programa em execução constitui um processo. Considerando um ambiente multiprocessado, podemos ter processos em inúmeros processadores.

Observação:

O MPI Padrão estipulou que a quantidade de processos deve corresponder à quantidade de processadores disponíveis.

Mensagem (*message*):

É o conteúdo de uma comunicação, formado de duas partes:

- ♦ Envelope:

Endereço (origem ou destino) e rota dos dados. O envelope é composto de três parâmetros:

- ♦ Identificação dos processos (transmissor e receptor);

- ◆ Rótulo da mensagem;
 - ◆ Comunicator.
- ◆ Dado:
- Informação que se deseja enviar ou receber. É representado por três argumentos:
- ◆ Endereço onde o dado se localiza;
 - ◆ Número de elementos do dado na mensagem;
 - ◆ Tipo do dado.

Rank:

Todo o processo tem uma identificação única atribuída pelo sistema quando o processo é inicializado. Essa identificação é contínua representada por um número inteiro, começando de zero até N-1, onde N é o número de processos.

Rank é o identificador único do processo, utilizado para identificar o processo no envio (send) ou recebimento (receive) de uma mensagem.

Group:

Group é um conjunto ordenado de N processos. Todo e qualquer group é associado a um communicator muitas vezes já predefinido como "MPI_COMM_WORLD". Inicialmente, todos os processos são membros de um group com um communicator.

Communicator:

O communicator é um objeto local que representa o domínio (contexto) de uma comunicação (conjunto de processos que podem ser contactados).

O MPI utiliza esta combinação de grupo e contexto para garantir segurança na comunicação e evitar problemas no envio de mensagens entre os processos.

A maioria das rotinas de PVM exigem que seja especificado um communicator como argumento. O MPI_COMM_WORLD é o comunicador predefinido que inclui todos os processos definidos pelo usuário numa aplicação MPI.

Application Buffer:

É o endereço de memória, gerenciado pela aplicação, que armazena um dado que o processo necessita enviar ou receber.

System Buffer:

É um endereço de memória reservado pelo sistema para armazenar mensagens.

Dependendo do tipo de operação de envio/recebimento (send/receive), o dado no buffer da aplicação (application buffer) pode necessitar ser copiado de/para o buffer do sistema (system buffer) através das rotinas send buffer/receive buffer. Neste caso a comunicação é assíncrona.

Blocking Communication:

Em uma rotina de comunicação blocking, a finalização da chamada depende de certos eventos.

Em uma rotina de envio de mensagem, o dado tem que ter sido enviado com sucesso, ou ter sido salvo no buffer do sistema (system buffer), indicando que o endereço do buffer da aplicação (application buffer) pode ser reutilizado.

Em uma rotina de recebimento de mensagem, o dado tem que ser armazenado no buffer do sistema (system buffer), indicando que o dado pode ser utilizado.

Non-Blocking Communication:

Em uma rotina de comunicação non-blocking, a finalização da chamada não espera qualquer evento que indique o fim ou sucesso da rotina.

As rotinas non-blocking communication não esperam pela cópia de mensagens do buffer da aplicação (application buffer) para o buffer do sistema (system buffer), ou a indicação do recebimento de uma mensagem.

Overhead:

Overhead é um desperdício de tempo que ocorre durante a execução dos processos. Os fatores que levam ao overhead são: comunicação, tempo em que a máquina fica ociosa (tempo idle) e computação extra.

Existem os seguintes tipos de overhead:

◆ *System Overhead*

É o tempo gasto pelo sistema na transferência dos dados para o processo destino.

◆ *Synchronization Overhead*

É o tempo gasto para a sincronização dos processos.

Observação:

O MPI sempre apresenta um overhead de sincronização inicial, quando espera até que todas as máquinas estejam disponíveis para iniciar o processamento e inicializar os processos. O mesmo ocorre quando da conclusão: existe um delay até que todos os

processos possam encerrar adequadamente e terminarem a execução.

2.12.2. Utilizando o MPI

O primeiro passo para a utilização do MPI é incluir no código fonte a biblioteca MPI. Para programas em C utiliza-se a biblioteca **mpi.h** e para fortran **mpif.h**.

O passo seguinte é escrever o programa fazendo chamadas às rotinas MPICH (versões gratuita do MPI) e compilar com a sintaxe adequada, incluindo os parâmetros desejados.

Para a compilação e linkedição utilizar os comandos para:

Linguagem C (C++)

`mpicc [fonte.c] -o [executável] [parâmetros]`

O comando `mpicc` aceita todos os argumentos de compilação do compilador C .

FORTRAN

`mpif77 [fonte.f] -o [executável] [parâmetros]`

`mpif90 [fonte.f90] -o [executável] [parâmetros]`

O comando `mpif77` e `mpif90` aceita todos os argumentos de compilação do compilador Fortran da Intel (`ifc`).

Para execução de programas paralelos basta utilizar o comando **mpirun** para linguagens C e FORTRAN:

`mpirun -[argumentos] [executável]`

Argumentos:

- `h` - Mostra todas as opções disponíveis
- `arch` - Especifica a arquitetura da(s) máquina(s)
- `machine` - Especifica a(s) máquina(s)
- `machinefile` - Especifica o arquivo que contém o nome das máquinas
- `np` - Especifica o número de processadores
- `leave_pg` - Registra onde os processos estão sendo executados
- `nolocal` - Não executa na máquina local
- `t` - Testa sem executar o programa, apenas imprime o que será executado
- `dbx` - Inicializa o primeiro processo sobre o `dbx`

Exemplos:

```
mpirun -np 5 teste
```

Executa o programa teste em 5 processadores

Em locais de arquitetura mista, é possível mesclar os parâmetros, de forma a executar sob o MPI, um mesmo programa compilado diferentemente nas várias arquiteturas.

Exemplo:

```
mpirun -arch risc6000 -np 3 -arch sun4 -np 2 teste.%a onde %a substitui a arquitetura.
```

2.13. Implementações

Para começar a desenvolver programas paralelos, uma série de etapas foram necessárias a fim de se obter uma base sólida de conhecimento. Testes de performance foram realizados e vários programas implementados. Será destaque neste tópico, duas implementações.

2.14. Processamento paralelo e CFD

Para o projeto de um algoritmo paralelo, a lei de Amdahl diz que para se obter uma alta eficiência, a porção do código que não pode ser paralelizado deve ser muito pequena.

Como o LTCM trabalha com CFD (Computational Fluids Dinamic), a melhor forma de implementar o algoritmo paralelo é subdividir o domínio da solução em subdomínios e, assim, determinar cada subdomínio para cada processador. Neste caso, o mesmo código roda em todos os processadores e cada um deles executa cálculos sobre seu próprio conjunto de dados. Uma vez que cada processador precisa de dados que residem em outros subdomínios, a troca de dados entre os processadores é necessária.

Métodos explícitos são relativamente fáceis de paralelizar, visto que todas as operações são executadas sobre os dados do passo de tempo anterior. Neste caso, somente é necessário trocar os dados nas regiões de interface entre os subdomínios vizinhos assim que é completado cada passo de tempo. A sequência de operações e os resultados devem ser idênticos sobre um ou vários computadores. A parte mais difícil do problema é, normalmente, a solução da equação elíptica de Poisson para a pressão.

Métodos implícitos são mais difíceis de paralelizar. Enquanto o cálculo da matriz de coeficientes e do termo fonte usa somente dados da iteração anterior e podem ser

calculados eficientemente em paralelo, a solução do sistema de equações lineares não é fácil de paralelizar.

2.15. Bibliografia

CENAPAD-NE: Apostila do Workshop MPI. Fevereiro 1998

<http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=733>

<http://www.clubedohardware.com.br/super.html>