**Relatório Técnico**
**BOLSA DE PARTICIPAÇÃO EM CURSO OU ESTÁGIO TÉCNICO NO EXTERIOR – NÍVEL IV**

**NÚMERO DO PROCESSO FAPESP: 05/54944-3**
Período: 01/09/2005 a 01/10/2005

# TRANSFERÊNCIA DE TECNOLOGIAS PARA APLICAÇÃO DIRETA AO MONITORAMENTO E NOWCASTING USANDO OS RADARES METEOROLÓGICOS DOPPLER DO IPMET-UNESP

**JAQUELINE MURAKAMI KOKITSU**
**Coordenador: Dr. Gerhard Held**
**outubro/2005**

# Sumário

# 1. INTRODUÇÃO

A previsão imediata ou nowcasting compreende etapas como iniciação de tempestades, sua evolução e seu deslocamento. A maioria das técnicas desenvolvidas para a previsão imediata ou nowcasting da atividade convectiva incorporam de alguma forma o rastreamento, com a extrapolação do deslocamento dessas entidades (tempestades).

Uma excelente revisão a respeito dos modelos e das técnicas utilizadas em previsão imediata ou "nowcasting" usando radar meteorológico, pode ser encontrada em Collier (1996) e Wilson (1998); onde são discutidas as vantagens e desvantagens dos vários métodos empregados, tais como, o método do centróide, o da correlação cruzada e outros mais complexos que fazem uso de, por exemplo, transformadas de Fourier.

No início dos anos 90, pesquisadores do National Center for Atmospheric Research (NCAR), em Boulder, Estados Unidos, refinaram e melhoraram um sistema desenvolvido para aplicação em previsão imediata do deslocamento de tempestades, baseado na metodologia de centróides, denominado TITAN (Thunderstorm, Identification, Tracking, Analysis and Nowcasting), Dixon e Wiener (1993). Este método define as tempestades como regiões tridimensionais de refletividades excedendo um determinado limiar e combinando-as de modo lógico, entre duas observações consecutivas de radar. O método usa como base as informações do radar em coordenadas cartesianas. A componente de rastreamento está baseada na solução otimizada do problema de "matching", e não na hipótese sobre a velocidade inicial da tempestade. Fusões e divisões –"merger" e "split" - são identificadas através de lógica geométrica considerando as posições e formas das tempestades. Por fim, as previsões são baseadas no ajuste linear considerando a história da tempestade em relação às suas posições e formas. O sistema foi projetado para funcionar em tempo real, com dados de radar, provendo a análise e a previsão num tempo aproximado de 10 segundos a partir do término de coleta da varredura volumétrica.

Adaptações e mudanças são necessárias no software TITAN para atender as necessidades do projeto SIHESP (SISTEMA INTEGRADO DE HIDROMETEOROLOGIA DO ESTADO DE SÃO PAULO), cuja integração dos radares meteorológicos do sistema está sob a responsabilidade do IPMet, assim como o monitoramento e previsão imediata de tempestades na área do Estado de São Paulo. É

importante destacar ainda que os dois radares Doppler do IPMet terão seus sistemas de processamento, coleta, geração de produtos e arquivamento de dados atualizados através do Projeto FAPESP no. 01/14095-6, outorgado em janeiro de 2005.

O treinamento realizado no Research Applications Laboratory (RAL) do NCAR, Boulder-CO, tem como objetivo principal, o conhecimento dos diferentes módulos (programas) que compõem o software TITAN bem como a interação entre eles de modo a possibilitar a adaptação adequada do mesmo às necessidades do IPMet e do projeto SIHESP.

Este treinamento foi ministrado pelo engenheiro de software Michael Dixon e utilizou como material de apoio as seguintes referências bibliográficas: Dixon (1994), Titan Training Mendoza Province (2001). A nova documentação do sistema TITAN foi elaborada durante o período do treinamento e está disponível junto com o pacote de distribuição do software TITAN – Dixon (2005).

# 2. CARACTERÍSTICAS DO SISTEMA TITAN

O projeto TITAN teve início em 1982 com o objetivo de identificar e analisar tempestades para avaliação de atividades de indução artificial de chuvas por semeadura de nuvens na África do Sul. Com o passar dos anos o TITAN foi se transformando num grande sistema de software que, além de aplicação específica de identificação e previsão de tempestades de chuvas usando radares meteorológicos, suporta também a introdução de diversos tipos de dados, realiza processamentos e geração de novos produtos, apresenta resultados graficamente e disponibiliza dados para uso em outros sistemas.

O sistema TITAN apresenta hoje componentes que realizam as seguintes tarefas:

- introdução de dados de vários tipos de radares meteorológicos;
- introdução de outros tipos de dados como trajetória de aviões, descargas atmosféricas, satélite, modelos numéricos, estações meteorológicas;
- remapeamento de dados de radar em coordenadas cartesianas;
- composição de radares;

- identificação e remoção de ecos de terreno e propagações anômalas em dados de radares;
- rastreamento e previsão de tempestades;
- estimativa de precipitação;
- processamento de VIL e severidade de tempestades.

A primeira parte do treinamento se concentrou na apresentação de aspectos básicos do sistema TITAN, como a compilação e a instalação do software, os tipos de dados internos utilizados, a estrutura de diretórios do sistema e os principais programas executados.

## 2.1 Download e Compilação

Os direitos de propriedade intelectual do TITAN pertencem ao UCAR (University Corporation for Atmospheric Research). A página web para aquisição do software está localizada em www.rap.ucar.edu/projects/titan. A distribuição do código fonte é feita através de arquivo do tipo "tar" compactado ("gzip"), com o seguinte formato de nome: titan5.20050922.src.tgz. O sistema operacional LINUX é o mais comumente utilizado para executar o TITAN, bem como o ambiente shell csh. Algumas bibliotecas e pacotes são necessários para compilar o software: compilador gcc/g++ 3.0 ou superior, X11R6, perl, bibliotecas do ImageMagick (imlib, gif, jpeg, png, tiff), entre outros.

Para descompactar o TITAN numa máquina LINUX é necessário executar os seguintes comandos:

```
mkdir ~/rap
cp titan5.20050922.src.tgz ~/rap
cd ~/rap
tar xvfz titan5.20050922.src.tgz
```

Após a descompactação, a estrutura de diretórios apresentada é a seguinte:

```
~/rap/bin (programas executáveis em formato binário e "scripts")
~/rap/lib (bibliotecas)
~/rap/lib/perl5 (módulos perl)
~/rap/include (arquivos "include")
```

```
~/rap/make_include (arquivos "include" para os "Makefiles")
```

Para que o sistema TITAN possa ser compilado e executado é necessário que algumas variáveis de ambiente estejam configuradas. A forma mais simples de se obter o correto ambiente para execução do TITAN é através da cópia do arquivo .cshrc de um dos diretórios de modelos de projetos TITAN:

```
cd ~/rap/projects/titan/templates/template_single_radar/system/dotfiles
cp cshrc ~/.cshrc
cd
source .cshrc
```

O script build_titan deve ser executado para compilação do pacote:

```
cd ~/rap
build_titan
```

## 2.2 Configuração de um Projeto

O próximo passo consiste na criação de diretórios, programas e arquivos de configuração de um "projeto". O projeto é a configuração do TITAN para execução em tempo real ou pós-facto. Cada usuário deve desenvolver um projeto adequado às suas necessidades. Nesta etapa de configuração do TITAN devem ser definidos os dados que serão introduzidos no sistema e quais os produtos que devem ser gerados pelo sistema para análise e visualização.

Uma convenção adotada pelo TITAN é a utilização do diretório ~/projDir para armazenamento de programas e arquivos do projeto. A forma mais simples de desenvolvimento de um projeto é a partir de um modelo já pronto. A distribuição do TITAN já possui alguns modelos ou exemplos de projetos. Para instalar um exemplo de projeto basta executar:

```
cd ~/rap/projects/titan/templates/template_single_radar
cd system/scripts
./INSTALL_TITAN
cd
source ~/.cshrc
```

O diretório ~/projDir, também referenciado pela variável $PROJ_DIR, possui normalmente a seguinte estrutura:

```
projDir/
  control/ (armazena a lista de processos em execução e a tabela 'crontab')
  data/    (diretório para os dados, muitas vezes é um link simbólico)
    raw/
    mdv/
    spdb/
    fmq/
    titan/
  logs/    (diretório para os logs do sistema)
    errors/
    restart/
  system/  (armazena scripts e arquivos de parâmetros gerais do sistema)
    scripts/
    params/
  ingest/  (armazena scripts e arquivos de parâmetros para introdução de dados)
    scripts/
    params/
  titan/   (armazena scripts e arquivos de parâmetros para os processos do titan)
    scripts/
    params/
  display/ (diretório com arquivos e scripts para visualização)
    scripts/
    params/
    maps/
    color_scales/
```

## 2.3 Tipos de Dados Internos

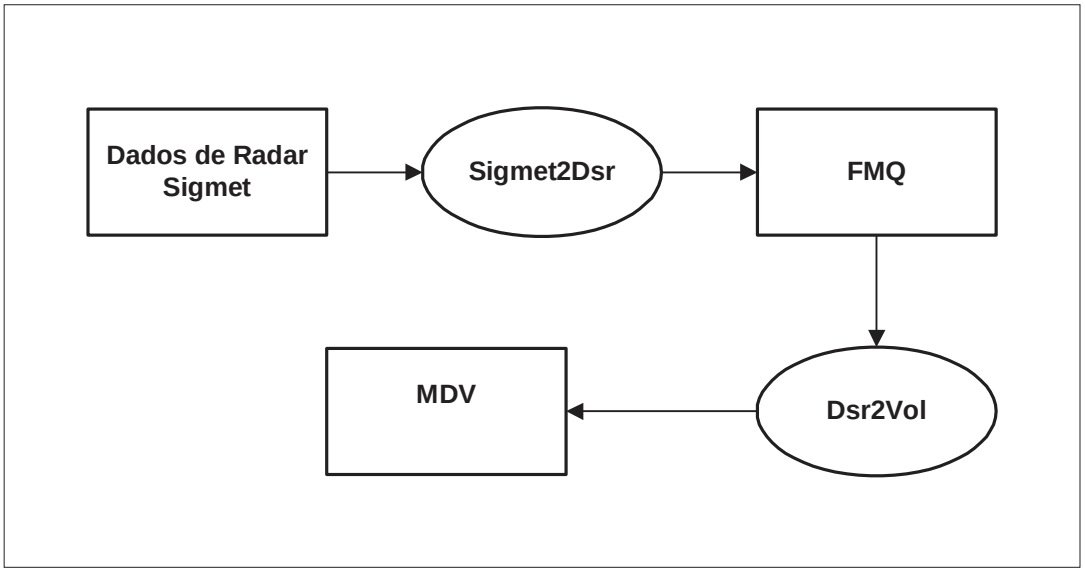
Os tipos de dados internos suportados pelo TITAN são:

- MDV (Meteorological Data Volume) para dados no formato de grade. O formato MDV para dados na forma de grade foi desenvolvido no NCAR no início dos anos 90. O MDV é um formato capaz de armazenar dados em grade de até três dimensões. Ele é altamente estruturado e possui habilidade para gerenciar múltiplos campos de dados em um único arquivo. O MDV requer o espaçamento constante de dados nos planos x-y para cada campo, ou seja, um único delta-x e delta-y para todos os dados de um determinado campo. Entretanto, o delta-x e o delta-y podem variar de campo para campo. Na terceira dimensão, o espaçamento poder ser variável, aceitando no máximo 122 níveis verticais.

- SPDB (Symbolic Product Data Base) para dados pontuais como trajetória de aviões, dados de estações meteorológicas, dados de descargas atmosféricas (raios), etc;
- TITAN para dados de tempestades e de rastreamento de tempestades;
- FMQ (File Queue Message) para dados de armazenamento temporário em uma fila do tipo “First In First Out” (FIFO).

A introdução de dados no sistema TITAN requer a execução de programas conversores de formato. O processo de conversão de dados brutos de radar armazenados por feixes (beam-by-beam) para o formato MDV do TITAN é realizado através de duas etapas. Na primeira etapa é executado o programa para converter o dado original para o formato FMQ. Na segunda etapa um outro programa é executado para ler o formato FMQ e gerar o dado no formato MDV.

Um esquema básico de introdução de dados do radar Sigmet de Bauru no sistema TITAN pode ser observado abaixo:




Os dados de radar no formato original (Sigmet) devem ser armazenados no diretório ~/projDir/data/raw/sigmet/bauru. A variável de ambiente $DATA_DIR especifica o diretório ~/projDir/data. O programa Sigmet2Dsr é executado para converter os dados no formato Sigmet para o formato FMQ. O dados nesse formato ficam em $DATA_DIR/fmq.

O programa Dsr2Vol é utilizado para converter o formato FMQ em MDV. Os arquivos no formato MDV ficam em $DATA_DIR/mdv.

## 2.4 TITAN: Aplicações

O sistema TITAN engloba um grande número de aplicações que executam desde a introdução de dados, a execução de algorítmos para geração de novos produtos, até a análise e visualização. A maioria dessas aplicações foi projetada para execução em dois modos: REALTIME, onde os processos são executados automaticamente assim que novos dados chegam, e ARCHIVE, onde a análise é realizada algum tempo após a coleta do dado.

Essas aplicações lêem os dados num formato, realizam algum processamento neles e escrevem num outro formato para serem lidos por outra aplicação.

A maioria das aplicações TITAN é configurada para utilizar um arquivo de parâmetros específicos para aquela aplicação. Quase todas as aplicações ajustam-se a três convenções na linha de comando:

- -h : mostra como executar a aplicação;
- -params: especifica o nome do arquivo de parâmetro a ser usado para rodar a aplicação;
- -print_params: mostra um modelo padrão do arquivo de parâmetros da aplicação na saída padrão (stdout);

A criação do arquivo de parâmetros de uma aplicação pode ser realizada da seguinte forma:

```
Sigmet2Dsr -print_params > Sigmet2Dsr.exemplo
```

ou

```
Sigmet2Dsr -print_params long > Sigmet2Dsr.exemplo
```

Este comando irá criar o arquivo Sigmet2Dsr.exemplo com parâmetros de configuração da aplicação Sigmet2Dsr já configurados com valores padrões. O segundo exemplo apresenta um arquivo com comentários mais completos. O usuário deve editar este arquivo para adequá-lo às suas necessidades.

Quando uma nova versão do TITAN é disponibilizada pode ser necessário atualizar o arquivo de parâmetros com as novas inclusões. Para aproveitar um arquivo de parâmetros antigo na criação de um novo atualizado deve-se proceder da seguinte maneira:

```
Dsr2Vol –params Dsr2Vol.old –print_params > Dsr2Vol.new
```

A execução da aplicação em tempo real deve ser realizada da seguinte maneira:

```
Sigmet2Dsr –params Sigmet2Dsr.exemplo –mode REALTIME >& /tmp/log.Titan &
```

No modo ARCHIVE o programa Sigmet2Dsr deve ser executada como segue:

```
Sigmet2Dsr -debug -params Sigmet2Dsr.bauru –mode ARCHIVE -f \
~/projDir/data/raw/sigmet/bauru/BRU040525070124.RAW8M9U \
~/projDir/data/raw/sigmet/bauru/BRU040525070830.RAW8M9V \
~/projDir/data/raw/sigmet/bauru/BRU040525071625.RAW8M9X
```

As aplicações que trabalham com dados internos do TITAN podem ser executadas em modo ARCHIVE usando o parâmetro –start e –end. O programa Titan pode ser executado com o comando:

```
Titan –param Titan.test –mode ARCHIVE –start “2005 09 21 00 00 00” \
–end “2005 09 22 00 00 00”
```

## 2.5 TITAN: Tempo Real

O sistema TITAN em tempo real engloba diversos componentes que, funcionando em conjunto, iniciam e param o sistema, mantém o sistema em execução contínua, monitoram processos, dados, espaço em disco, erros, entre outros.

Alguns componentes importantes do sistema TITAN em tempo real são os arquivos contidos no diretório $PROJ_DIR/control:

- **proc_list**: lista os processos a serem executados;

- **crontab**: especifica os programas a serem executados automaticamente pelo sistema operacional LINUX.

A seguir é apresentado um exemplo do arquivo proc_list que especifica os processos a serem executados pelo sistema TITAN, como por exemplo, os programas de controle do sistema, os programas de introdução de dados de radares Sigmet, programas para realizar a remoção de ecos de terreno, programa para realizar a composição dos radares, processos do TITAN e programas de visualização. O proc_list especifica o nome do processo, a instância e o nome dos scripts de inicialização e finalização do processo e o nome da máquina que executa o processo.

```
##############################################################################
# SYSTEM processes
#
DsServerMgr   primary    start_DsServerMgr         snuff_inst       localhost
Janitor       primary    start_Janitor             kill_Janitor     localhost
Scout         primary    start_Scout               kill_Scout       localhost
DataMapper    primary    start_DataMapper          kill_DataMapper  localhost
##############################################################################
# INGEST SIGMET radars, Remove Clutter and Merging
#
Sigmet2Dsr    bauru      start_Sigmet2Dsr.bauru       snuff_inst  localhost
Dsr2Vol       bauru      start_Dsr2Vol.bauru          snuff_inst  localhost
ClutterRemove bauru      start_ClutterRemove.bauru    snuff_inst  localhost

#
Sigmet2Dsr    prudente   start_Sigmet2Dsr.prudente    snuff_inst  localhost
Dsr2Vol       prudente   start_Dsr2Vol.prudente       snuff_inst  localhost
ClutterRemove prudente   start_ClutterRemove.prudente snuff_inst  localhost
#
# Merge the radars
MdvMerge2     ops        start_MdvMerge2.ops          snuff_inst  localhost
##############################################################################
# TITAN PROCESSES
#
Titan           merged   start_Titan.merged           snuff_inst  localhost
PrecipAccum     single   start_PrecipAccum.single     snuff_inst  localhost
PrecipAccum     1hr      start_PrecipAccum.1hr        snuff_inst  localhost
```

```
PrecipAccum     24hr     start_PrecipAccum.24hr       snuff_inst  localhost
Mdv2Vil         merged   start_Mdv2Vil.merged         snuff_inst  localhost
Tstorms2Spdb    merged   start_Tstorms2Spdb.merged    snuff_inst  localhost
StormInitDetect merged   start_StormInitDetect.merged snuff_inst  localhost
#############################################################################
# DISPLAY processes
#
Rview        merged     start_Rview.merged     snuff_inst          localhost
TimeHist     merged     start_Rview.merged     snuff_inst          localhost
RadMon       merged     start_RadMon.merged    kill_RadMon.merged  localhost
CIDD         merged     start_CIDD.merged      snuff_inst          localhost
```

O sistema TITAN está baseado no uso de servidores de dados, que são aplicações que recebem requisições de outros programas (clientes) e respondem ao pedido lendo ou escrevendo dados em algum dispositivo de armazenamento. Programas servidores normalmente “escutam” alguma porta virtual. Quando uma aplicação cliente conecta-se a esta porta, o servidor cria uma imagem de si mesmo para gerenciar o pedido ou requisição. Dessa forma o servidor fica disponível para o atendimento a outros pedidos. Um dos principais servidores de dados é o programa DsServerMgr, que é responsável pela inicialização de outros servidores quando necessário. Se um programa cliente tenta acessar um servidor que não responde, ele contacta o DsServerMgr, que irá reiniciar o programa servidor requisitado e avisar o cliente para este tentar a conexão novamente.

O arquivo crontab do UNIX é responsável pela execução automática de processos no sistema operacional. Abaixo é mostrado um arquivo crontab típico do TITAN para execução em tempo real:

```
######################################################################
# SYSTEM
#
# Process restarters
*/1 * * * *  csh -c "start_auto_restart_check_cron" 1> /dev/null 2> /dev/null
*/1 * * * *  csh -c "start_procmap_check_cron"      1> /dev/null 2> /dev/null
#
# Build links to log date subdirs
*/5 * * * *   csh -c "start_build_logdir_links" 1> /dev/null 2> /dev/null
#
```

Este arquivo mostra três tarefas escalonadas:

- A cada 1 minuto o script start_auto_restart_check_cron é executado para assegurar que o script auto_restart esta executando.
- A cada 1 minuto o script start_procmap_check_cron é executado para verificar se o processo procmap está rodando.
- A cada 5 minutos o script start_build_logdir_links roda para criar links simbólicos no diretório de logs para apontar os arquivos de log de ontem e de hoje.

O processo ‘procmap’ é o responsável pela capacidade de re-inicialização do sistema TITAN. Ele mantém uma tabela com o status de todos os processos em execução no sistema. Cada processo em execução registra informações na tabela do procmap a cada 1 minuto. Esta tabela é lida pelo processo procmap e comparada com a lista de processos contida no arquivo proc_list. Se algum processo está faltando ele é automaticamente re-iniciado.

O processo ‘DataMapper’ realiza tarefa similar ao procmap em relação aos dados no sistema TITAN. Cada vez que uma aplicação escreve dados no disco ela registra esta atividade na tabela do ‘DataMapper’.

O script ‘auto_restart’ é responsável por contactar o processo ‘procmap’ em intervalos regulares de tempo e checar a tabela de processos em execução comparando-a com a lista de processos contida no arquivo proc_list.

Existem dois tipos de arquivos de logs: os arquivos de erros, que ficam em $PROJ_DIR/logs/errors, e os arquivos de re-inicialização de processos, que ficam em $PROJ_DIR/logs/restart. Os arquivos de logs residem em diretórios nomeados por data. Os logs de erros contêm mensagens de erros geradas pelas aplicações. Por exemplo, a aplicação PrecipAccum rodando com a instância 24hr irá criar o log denominado PrecipAccum.24hr.log.

A aplicação Janitor é usada para controlar o espaço em disco ocupado pelos dados do sistema TITAN. O Janitor opera em três principais funções: remove arquivos que ultrapassam determinada data, remove diretórios vazios e compacta arquivos após uma data especificada. A aplicação Janitor é configurada pelo arquivo _Janitor que normalmente é colocado no diretório $DATA_DIR. Os subdiretórios de $DATA_DIR podem conter arquivos _Janitor configurados de forma diferente. As configurações nos diretórios de níveis inferiores sobrepõem àquelas do nível superior.

O programa Scout possui propriedades semelhantes ao Janitor, só que ao invés de remover arquivos ou compactá-los, ele percorre os diretórios para obter informações sobre os dados e registrá-las no DataMapper.

Os scripts start_all e stop_all são responsáveis pelo início e finalização do sistema TITAN em tempo real. O script start_all realiza as seguintes tarefas:

1. Põe o processo procmap em execução
2. Inicia a execução de todos os processos listados no arquivo proc_list
3. Executa o script auto_restart
4. Põe o arquivo crontab em execução

Os scripts snuff e snuff_inst são usados para parar a execução de processos individuais. Para finalizar a execução de um processo, é necessário executar: snuff nome_processo ou snuff_inst nome_processo instancia

Para visualizar a lista de todos os processos em execução no sistema, digitar o comando ppm. O ppm é um apelido para o comando: "`print_procmap -hb -up -status`".

Abaixo é apresentado um exemplo da saída do comando ppm:

```
PROCS REGISTERED - localhost - Fri Oct 14 13:53:21 2005
Uptime: 8.91 d
```

| Name | Instance | Host | User | Pid | Heartbeat | Uptime | Status |
|---|---|---|---|---|---|---|---|
| ClutterRemov | bauru | yradar | titan | 4338 | 0:0:7 | 8.91 d | DsMdvxTimes::getNext zzzz |
| ClutterRemov | prudente | yradar | titan | 4381 | 0:0:7 | 8.91 d | DsMdvxTimes::getNext zzzz |
| DataMapper | primary | yradar | titan | 4295 | 0:0:47 | 8.91 d | Listening, port: 5434 |
| DsServerMgr | primary | yradar | titan | 4193 | 0:0:49 | 8.91 d | Listening, port: 5435 |
| Dsr2Vol | bauru | yradar | titan | 4323 | 0:0:5 | 8.91 d | In FMQ_read_blocking() |
| Dsr2Vol | prudente | yradar | titan | 4366 | 0:0:5 | 8.91 d | In FMQ_read_blocking() |
| GenPt2Sympro | manager | yradar | titan | 1518 | 0:0:46 | 8.99 d | Listening, port: 5465 |
| Janitor | primary | yradar | titan | 12768 | 0:0:12 | 4:22:31 | Sleeping between passes |
| Ltg2Symprod | manager | yradar | titan | 20919 | 0:0:12 | 6.93 d | Listening, port: 5450 |
| Mdv2Vil | merged | yradar | titan | 4465 | 0:0:14 | 8.91 d | LdataInfo::readBlocking |
| MdvMerge2 | ops | yradar | titan | 4395 | 0:0:33 | 8.91 d | DsMdvxTimes::getNext |
| PrecipAccum | 1hr | yradar | titan | 4437 | 0:0:14 | 8.91 d | LdataInfo::readBlocking |
| PrecipAccum | 24hr | yradar | titan | 4451 | 0:0:13 | 8.91 d | LdataInfo::readBlocking |
| PrecipAccum | single | yradar | titan | 4423 | 0:0:14 | 8.91 d | LdataInfo::readBlocking |
| Scout | primary | yradar | titan | 4261 | 0:0:37 | 8.91 d | Sleeping between runs |
| Sigmet2Dsr | bauru | yradar | titan | 10283 | 0:0:10 | 8.89 d | DSINP_next: waiting for files |
| Sigmet2Dsr | prudente | yradar | titan | 4352 | 0:0:40 | 8.91 d | DSINP_next: waiting for files |
| StormInitDet | merged | yradar | titan | 4493 | 0:0:41 | 8.91 d | Waiting for new data... |
| Titan | merged | yradar | titan | 4409 | 0:0:33 | 8.91 d | DsMdvxTimes::getNext |
| Tstorms2Spdb | merged | yradar | titan | 4479 | 0:0:13 | 8.91 d | LdataInfo::readBlocking |
| Tstorms2Symp | manager | yradar | titan | 20831 | 0:0:24 | 6.93 d | Listening, port: 5460 |

A visualização do conjunto de dados disponível no sistema TITAN pode ser obtida através da execução do comando pdm, que é apelido para o comando:

```
PrintDataMap –all –relt –lreg
```

Abaixo, um exemplo da saída do comando pdm:

========== Data on host 'localhost' at time 2005/10/17 15:01:40 ==========

| DataType | Dir | HostName | Latest | Last reg | Start date | End date | nFiles | nBytes |
|---|---|---|---|---|---|---|---|---|
| mdv | mdv/no_clutter/bauru | pcpesq11 | +00:50:24 | -00:00:10 | 2005/10/17 | 2005/10/17 | 39 | 9.4M |
| mdv | mdv/no_clutter/merged | pcpesq11 | +00:51:35 | -00:03:05 | 2005/10/17 | 2005/10/17 | 11 | 4.1M |
| mdv | mdv/no_clutter/prudente | pcpesq11 | +00:51:35 | -00:03:11 | 2005/10/17 | 2005/10/17 | 12 | 3.4M |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| mdv | mdv/precip/1hr | pcpesq11 | +00:51:35 | -00:03:00 | 2005/10/17 | 2005/10/17 | 11 | 2.6M |
| mdv | mdv/precip/24hr | pcpesq11 | +00:51:35 | -00:03:00 | 2005/10/17 | 2005/10/17 | 11 | 3.4M |
| mdv | mdv/precip/single | pcpesq11 | +00:51:35 | -00:03:01 | 2005/10/17 | 2005/10/17 | 11 | 407K |
| mdv | mdv/radar/cart/bauru | pcpesq11 | +00:50:24 | -00:00:12 | 2005/09/19 | 2005/10/17 | 63 | 11M |
| mdv | mdv/radar/cart/prudente | pcpesq11 | +00:51:35 | -00:03:14 | 2005/09/19 | 2005/10/17 | 52 | 5.5M |
| mdv | mdv/radar/polar/bauru | pcpesq11 | +00:50:24 | -00:00:09 | 2005/10/17 | 2005/10/17 | 39 | 5.7M |
| mdv | mdv/radar/polar/prudente | pcpesq11 | +00:51:35 | -00:03:13 | 2005/10/17 | 2005/10/17 | 12 | 867K |
| mdv | mdv/radar/ppi/bauru | pcpesq11 | +00:50:24 | -00:00:11 | 2005/10/17 | 2005/10/17 | 39 | 4.0M |
| mdv | mdv/radar/ppi/prudente | pcpesq11 | +00:51:35 | -00:03:14 | 2005/10/17 | 2005/10/17 | 12 | 694K |
| mdv | mdv/vil/merged | pcpesq11 | +00:51:35 | -00:03:00 | 2005/10/17 | 2005/10/17 | 11 | 580K |
| spdb | spdb/StormInitLoc/merged | pcpesq11 | +00:51:35 | -00:03:01 | 2005/10/17 | 2005/10/17 | 2 | 9.3K |
| spdb | spdb/tstorms/merged | pcpesq11 | +00:51:35 | -00:03:02 | 2005/10/17 | 2005/10/17 | 2 | 32K |
| titan | titan/storms/merged | pcpesq11 | +00:51:35 | -00:03:02 | 2005/10/05 | 2005/10/17 | 6 | 347K |
| | | | | | | | ==== | ===== |
| | | | | | | | 333 | 52M |

# 3. O PROJETO IPMET

Esta etapa do treinamento caracterizou-se pela criação do ambiente TITAN em tempo real e pós-fato para o projeto IPMet. O projeto IPMet especifica a introdução automática de dados dos radares meteorológicos de Bauru e de Presidente Prudente, dados do satélite GOES, dados do modelo numérico de previsão do tempo ETA, dados das estações de METAR, radiossondas e estações de superfície. Também inclui a introdução em modo ARCHIVE de dados da trajetória de aviões e de descargas atmosféricas.

Em relação aos dados dos radares meteorológicos, o requerido é que eles passem pelo processo de remoção de “clutter” e que os dados dos radares de Bauru e de Presidente Prudente sejam compostos numa única imagem (“merging”). Além dos produtos de identificação e rastreamento de tempestades, gerados pelo programa Titan, também são requeridos os produtos: Precipitação Acumulada (1h e 24h), VIL (Vertical Integrated Liquid), DVIL (VIL Difference), SSS (Storm Severity Structure), Refletividade Máxima, Fluxo de Precipitação, Precipitação Total no Solo, Detecção de Início de Tempestade (Storm Init Detect), entre outros. Outra requisição é a ativação do processo de verificação da porcentagem de acerto da previsão realizada pelo Titan. Um programa de rastreamento e

previsão de chuva usando radar, denominado CTREC (Cartesian Tracking Radar Echoes by Correlation), também é desejado, para auxiliar na previsão da localização de todos os tipos de precipitação, não apenas as tempestades, usando o método da correlação cruzada.

Na questão da visualização o requerido é a apresentação de produtos de radar do TITAN nos visualizadores Rview e CIDD. O Rview á bastante apropriado para o acompanhamento do desenvolvimento das células de tempestades. O programa CIDD passará a ser um sistema de visualização capaz de integrar todos os dados recebidos e processados no IPMet, inclusive os do sistema TITAN, eliminando o uso de distintos programas executados em plataformas diferentes como ocorre atualmente no IPMet. O programa CIDD também deve gerar gráficos (jpg ou png) automaticamente para serem disponibilizados pela Internet.

## 3.1 Introdução de Dados do Radar Sigmet de Bauru

O format DSR-FMQ funciona como interface padrão entre o formato nativo de radar (no caso, o formato Sigmet) e o formato MDV do TITAN. A aplicação Sigmet2Dsr é responsável pela conversão do formato Sigmet para o formato FMQ. A conversão para o formato MDV é realizada pela aplicação Dsr2Vol. O volume de saída no formato MDV pode apresentar quatro possíveis tipos de coordenadas:

- Cartesiana: z em km MSL, y em km, x em km;
- PPI: z em graus de elevação, y em km, x em km;
- Polar: z em graus de elevação, y em graus de azimute e x em km de alcance;
- RHI: z em graus de azimute, x e y em graus de elevação.

As seguintes etapas são necessárias para a configuração do sistema TITAN de modo a permitir a introdução de dados do radar de Bauru.

1. Ir para o diretório $PROJ_DIR/ingest/params
2. Gerar o arquivo de parâmetros da aplicação Sigmet2Dsr excutando o comando:
   `Sigmet2Dsr –print_params >Sigmet2Dsr.bauru`

3. Editar o arquivo de parâmetros criado e adequá-lo as necessidades do projeto. Segue o exemplo de algumas alterações efetuadas:

```
> InDir = "/home/titan/projDir/data/raw/sigmet/bauru";
> radarName = "Bauru";
> instance = "bauru";
> output_fmq_url = "fmqp:://localhost::fmq/dsRadar.bauru";
```

   Obs.1: Os dados do radar de Bauru no formato Sigmet ficam em $DATA_DIR/raw/sigmet/bauru e os dados no formato FMQ ficam em $DATA_DIR/fmq.

   Obs.2: Este programa utiliza informações contidas no cabeçalho do arquivo de dados de radar no formato Sigmet.

4. Ir para o diretório $PROJ_DIR/ingest/scripts e criar o script shell start_Sigmet2Dsr.bauru:

```
#! /bin/csh -f

cd $PROJ_DIR/ingest/params

running "Sigmet2Dsr -params Sigmet2Dsr.bauru"
if ($status == 1) then
   Sigmet2Dsr -params Sigmet2Dsr.bauru |& \
   LogFilter -d $ERRORS_LOG_DIR -p Sigmet2Dsr -i bauru &
endif
```

5. Ir para o diretório $PROJ_DIR/ingest/params e gerar o arquivo de parâmetros da aplicação Dsr2Vol: Dsr2Vol –print_params >Dsr2Vol.bauru

6. Editar o arquivo de parâmetros Dsr2Vol.bauru. Alguns exemplos de parâmetros configuráveis:

```
> instance = "bauru";
> input_fmq_url = "fmqp:://localhost::fmq/dsRadar.bauru";
> check_min_beams_in_tilt = TRUE;
> min_beams_in_tilt = 90;
```

```
> check_min_fraction_in_tilt = TRUE;
> min_fraction_in_tilt = 0.25;
> sn_min_valid_run = 5;
> noise_dbz_at_100km = -10;
> bridge_missing_in_elevation = TRUE;
> min_nvalid_for_interp = 4;
>   {
>     dsr_name = "DZ",
>     output_name = "DBZ",
>     output_units = "dBZ",
>     transform = "dBZ",
>     is_dbz = TRUE,
>     interp_db_as_power = FALSE,
>     is_vel = FALSE,
>     allow_interp = TRUE,
>     encoding = ENCODING_INT8
>   }
>   {
>     dsr_name = "VR",
>     output_name = "VEL",
>     output_units = "m/s",
>     transform = "none",
>     is_dbz = FALSE,
>     interp_db_as_power = FALSE,
>     is_vel = true,
>     allow_interp = TRUE,
>     encoding = ENCODING_INT8
>   }
>   {
>     dsr_name = "SW",
>     output_name = "SPW",
>     output_units = "m/s",
>     transform = "none",
>     is_dbz = FALSE,
>     interp_db_as_power = FALSE,
>     is_vel = FALSE,
>     allow_interp = TRUE,
>     encoding = ENCODING_INT8
```

```
>   }
> output_cart_files = TRUE;
>     nxy = 640,
>     dxy = 0.75,
>     nz = 26,
>     minz = 2.0,
>     dz = 0.75,
>     interpolate = TRUE,
>     mdv_url = "$(DATA_DIR)/mdv/radar/cart/bauru"
>
> output_ppi_files = TRUE;
>     nxy = 480,
>     dxy = 1,
>     min_elev = 0,
>     max_elev = 90,
>     interpolate =TRUE,
>     mdv_url = "$(DATA_DIR)/mdv/radar/ppi/bauru",
>     min_ht = 0,
>     max_ht = 50
>
> output_polar_files = TRUE;
>     max_range = 400,
>     min_elev = 0,
>     max_elev = 90,
>     mdv_url = "$(DATA_DIR)/mdv/radar/polar/bauru"
```

Obs. Neste programa são definidos os tipos de coordenadas do volume MDV e a resolução do volume de dados na forma de grade. No caso do radar Bauru, os arquivos MDV são gerados nos formatos: Cartesiano, PPI e Polar. Para dados em coordenadas cartesianas e PPI, a aplicação está utilizando a opção de interpolação pelo método bilinear de 8 pontos (a outra opção disponível é a “vizinho mais próximo”). Para dados em coordenadas polares o azimute é arredondado para a unidade de resolução de azimute mais próxima, normalmente 1 grau. No caso do dado em coordenadas cartesianas o volume em grade gerado possui resolução 640x640x26 pontos nos eixos xyz, sendo de 0.75km o espaçamento da grade na horizontal e na vertical. Os dados em coordenadas PPI possuem grade de 480x480 pontos (x-y), com espaçamento de 1km, e

resolução vertical dependente do número de elevações da antena de radar. No formato Polar a grade possue 240x360 pontos (xy) e resolução vertical de 11 níveis (número de elevações da antena do radar).

7. Ir para o diretório $PROJ_DIR/ingest/scripts e criar o script shell star_Dsr2Vol.bauru:

```
#! /bin/csh -f

cd $PROJ_DIR/ingest/params

running "Dsr2Vol -params Dsr2Vol.bauru"
if ($status == 1) then
  Dsr2Vol -params Dsr2Vol.bauru |& \
    LogFilter -d $ERRORS_LOG_DIR -p Dsr2Vol -i bauru &
endif
```

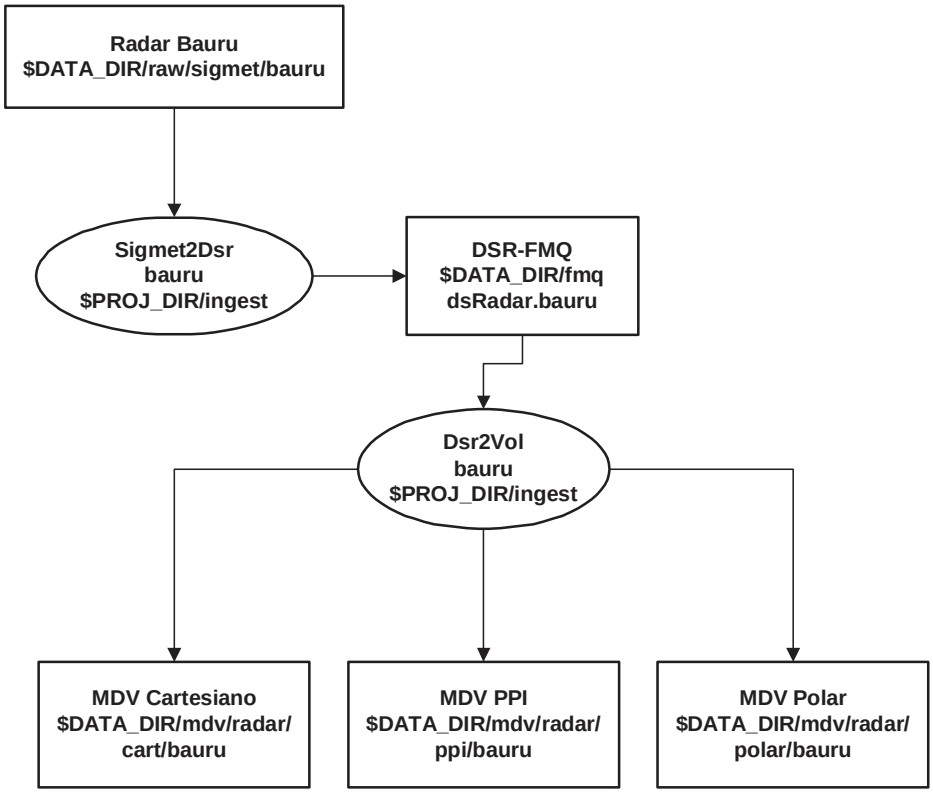
8. Ir para o diretório $PROJ_DIR/control e editar o arquivo proc_list, incluindo as linhas:

```
Sigmet2Dsr    bauru      start_Sigmet2Dsr.bauru     snuff_inst  localhost
Dsr2Vol       bauru      start_Dsr2Vol.bauru        snuff_inst  localhost
```

Dessa forma os processos de conversão de dados do radar Bauru em formato Sigmet para o formato MDV do TITAN serão executados automaticamente com a chegada de novos dados, após a execução do comando start_all. Observe que o arquivo proc_list deve conter algumas aplicações do sistema, como o DsServerMgr, o Janitor, o Scout e o DataMapper:

```
##############################################################################
# SYSTEM processes
#
DsServerMgr   primary    start_DsServerMgr         snuff_inst       localhost
Janitor       primary    start_Janitor             kill_Janitor     localhost
Scout         primary    start_Scout               kill_Scout       localhost
DataMapper    primary    start_DataMapper          kill_DataMapper  localhost
```

O fluxo de processos e dados da introdução de dados do radar Sigmet Bauru, incorporando informações da instância e a localização de diretórios, é apresentado a seguir:



## 3.2 Atenuação de Clutter

A técnica de remoção de clutter empregada no TITAN considera a geração do mapa de clutter ("clutter map") a partir de volumes em formato MDV (cartesiano) coletados em dias de céu claro, ou seja, dados sem a presença de chuva.

O programa MdvMedian é utilizado para computar os valores de refletividade médios para as varreduras de céu claro. Os arquivos gerados são armazenados como "clutter map" em formato MDV.

A aplicação ClutterRemove é usada para filtrar os dados de clutter dos volumes de radar contendo precipitação, usando dados dos arquivos "clutter map". Somente valores que excedam o valor médio serão mantidos.

1. O primeiro passo a ser realizado é a geração do arquivo MDV dos dados de céu claro coletados. Esta etapa foi apresentada na seção anterior, mas desta vez deve ocorrer no modo ARCHIVE e não no modo REALTIME:

   ```
   Sigmet2Dsr -params Sigmet2Dsr.bauru -f $DATA_DIR/raw/sigmet/bauru /*.*
   ```

   Considera-se que o diretório $DATA_DIR/raw/sigmet/bauru contém somente os arquivos de céu claro.

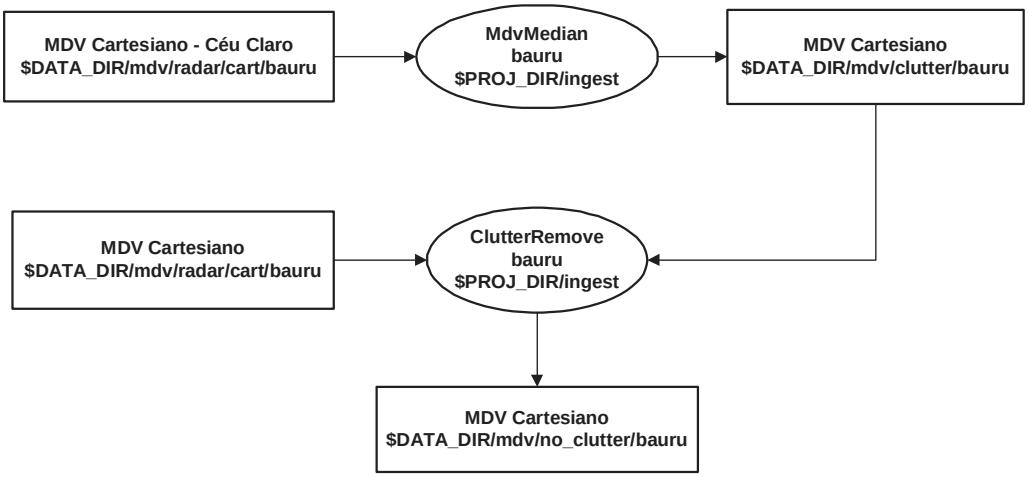
2. Executar a aplicação MdvMedian em modo ARCHIVE:

   ```
   MdvMedian -params MdvMedian.bauru -start "2005 09 19 10 00 00" \
   -end "2005 09 19 21 00 00"
   ```

3. Colocar processo ClutterRemove para execução automática no $PROJ_DIR/control/proc_list. Observar a criação do arquivo de parâmetros ClutterRemove.bauru e do scritp Start_ClutterRemove.bauru. O arquivo de parâmetros dessa aplicação deve ficar em $PROJ_DIR/ingest/params. Esse arquivo de parâmetros deve conter a variável use_latest_clutter_file configurada como TRUE, dessa forma o programa irá utilizar o último clutter map gerado em disco. Abaixo é apresentada a linha a ser incluída no arquivo proc_list:

   ```
   ClutterRemove bauru   start_ClutterRemove.bauru snuff_inst  localhost
   ```

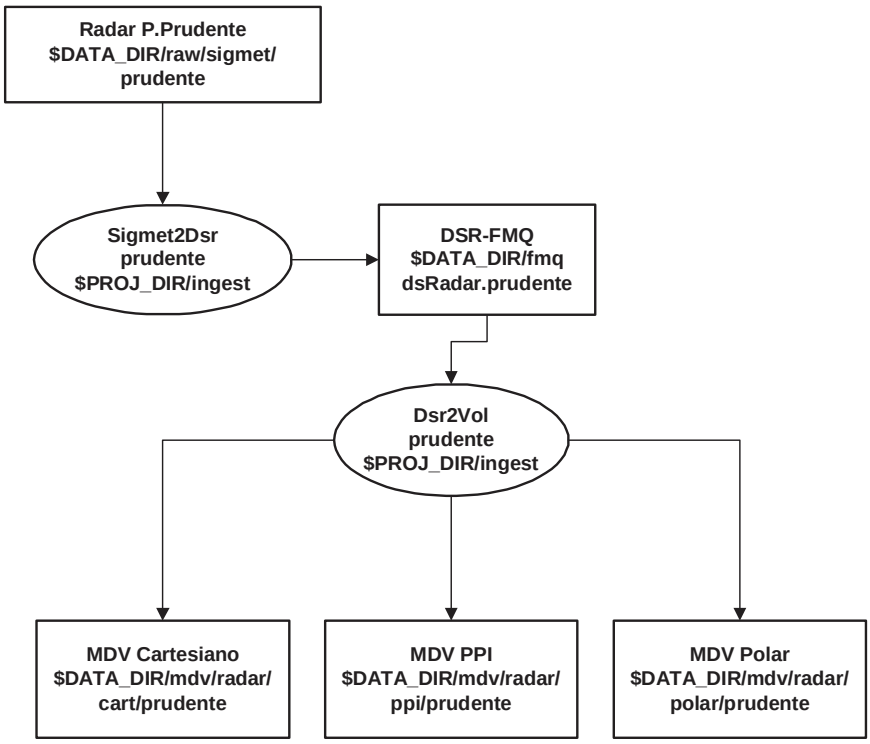
O fluxo de dados e processos pode ser observado na figura abaixo:



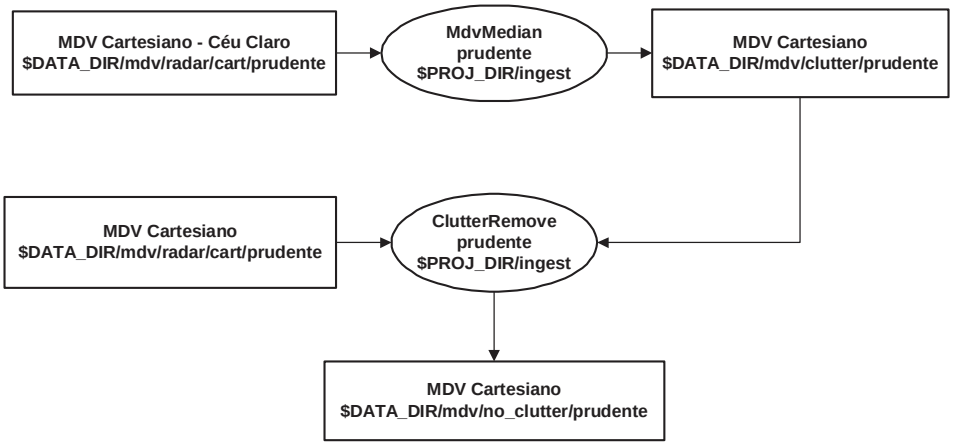
## 3.3 Introdução de Dados do Radar Sigmet de Presidente Prudente

Os processos de introdução de dados do radar de Presidente Prudente no sistema TITAN, bem como de remoção do clutter são similares aos realizados com os dados do radar de Bauru.

A seguir é apresentado o diagrama de fluxo de dados envolvidos na geração do arquivo MDV do radar de Presidente Prudente:



O fluxo de dados do processo de remoção de “clutter” dos dados de radar de Presidente Prudente pode ser observado abaixo:




## 3.4 Fusão de Múltiplos Radares (“Merging”)

No projeto IPMet os dados de radar são coletados por dois diferentes radares, um localizado em Bauru e outro em Presidente Prudente. Os dados de cada um desses radares podem ser combinados formando um único mosaico.

A aplicação MdvMerge2 permite a união de volumes de diferentes radares no formato de grade em um único mosaico comum. A projeção e a precisão na geração do mosaico são definidas no arquivo de configuração da aplicação. Os dados de entrada para essa aplicação deverão estar no formato de grade (MDV).

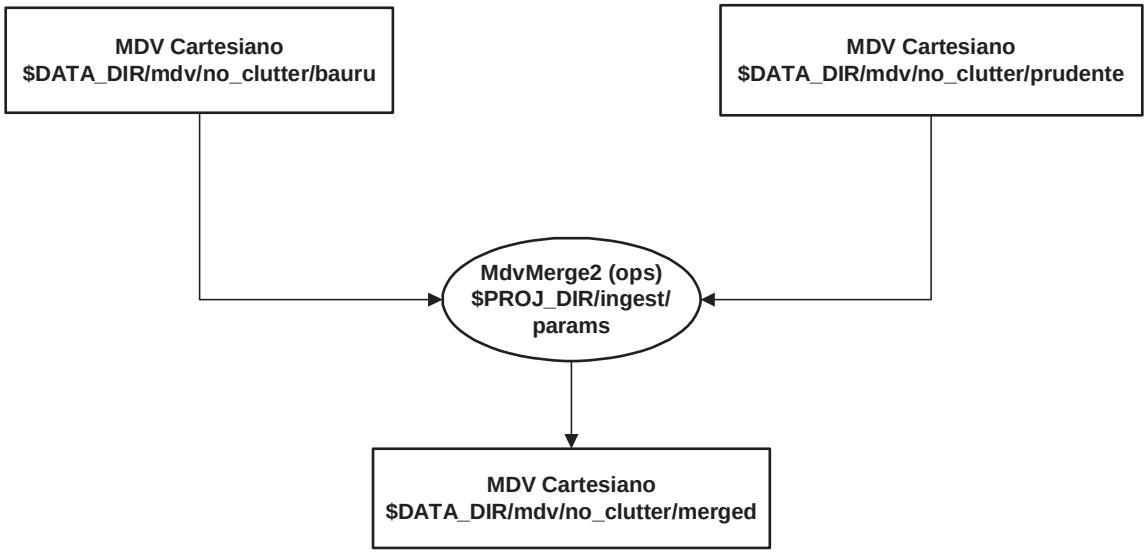
Uma das principais dificuldades no processo de “merging” é a decisão do momento de execução do “merging”. Uma das opções está baseada no acionamento da rotina em intervalo regulares de tempo. Na outra opção, que está sendo usada no projeto IPMet, a execução do “merging” ocorre quando um novo arquivo do primeiro radar listado (Prudente) for criado no disco. Uma amostra de alguns parâmetros do arquivo MdvMerge2.ops é apresentada a seguir.

```
> instance = "ops";
> trigger = FILE_TRIGGER;
>   { "DBZ", MERGE_MAX, FLOAT32, 0.5, -32, INT8},
>   { "VEL", MERGE_MEAN, FLOAT32, 0.5, -32, INT8},
>   { "SPW", MERGE_MEAN, FLOAT32, 0.5, -32, INT8}
>   { "mdvp:://localhost::mdv/no_clutter/prudente", "", TRUE, TRUE}
>   { "mdvp:://localhost::mdv/no_clutter/bauru", "", TRUE, TRUE},
> output_url = "mdvp:://localhost::mdv/no_clutter/merged";
> output_projection = OUTPUT_PROJ_LATLON;
> output_origin = { 0, 0 };
> output_grid = { 1000, 667, 26, -54.0, -24.5, 2, 0.0075, 0.0075, 0.75 };
> output_data_set_name = "IpMet Merged";
```

Nas regiões onde ambos os radares relatam dados, o maior valor de refletividade (DBZ) é usado para construir o mosaico, enquanto que para os campos de velocidade Doppler (VEL) e largura espectral (SPW) o valor médio é calculado.

O novo volume gerado possui dimensões 1000x667x26 (xyz), com espaçamento de 0.75 km entre os pontos de grade na horizontal e na vertical.

O “merging” dos radares de Bauru e Presidente Prudente pode ser observado no diagrama abaixo:



## 3.5 Aplicação Titan

A aplicação Titan identifica células de tempestades em dados volumétricos de radar em coordenadas cartesianas e verifica a trajetória das células identificadas, fornecendo uma previsão de seu movimento. Na configuração do projeto IPMet, essa aplicação utiliza volumes integrados dos radares de Bauru e de Presidente Prudente em formato MDV, formando uma grade tridimensional de 1000x667x26 (xyz), sendo o espaçamento desta grade de 0.75 km na horizontal e na vertical. A aplicação Titan identifica uma tempestade como sendo uma região contígua tridimensional de tal forma que certos parâmetros, como a refletividade, o volume, a altura, entre outros, satisfaçam determinadas condições. Na configuração do Titan para o projeto IPMet foram utilizados os valores de 40 DBZ de refletividade mínima, o volume mínimo de 16 $km^3$, e a altura mínima de 2 km e máxima de 30 km como parâmetros de identificação de uma tempestade.

O movimento das tempestades identificadas é determinado pelo uso de um método de otimização para calcular a melhor associação lógica entre tempestades ocorridas em duas observações consecutivas de radar. Este método assume o menor caminho entre as tempestades, aquelas com características similares (tamanho, forma, etc) e um limite máximo de distância que uma tempestade pode se mover num determinado intervalo de tempo para se encontrar a correta associação entre as tempestades. Essa combinação determina uma tendência no comportamento da tempestade, sobre a qual a previsão é realizada. Fusões e divisões de células de tempestades são identificadas através de lógica geométrica considerando as posições e formas das tempestades. Cada nova trajetória da tempestade está associada a dois números: um denominado complexo e outro simples. O número complexo identifica a tempestade original e o número simples indica fusão ou divisão da célula de tempestade. Por exemplo, na notação 751/1109, o valor 751 refere-se ao número complexo (tempestade original) e 1109 é o número simples.

Algumas características dos parâmetros de configuração da aplicação Titan atualmente em execução no IPMet estão listadas abaixo:

```
> instance = "merged";
> restart_time = { 13, 0 };
> restart_overlap_period = 10800;
> input_url = "mdvp:://localhost::mdv/no_clutter/merged";
> dbz_field = { "DBZ", 0 };
```

```
> vel_available = TRUE;
> low_dbz_threshold = 40;
> high_dbz_threshold = 1000;
> base_threshold = 2;
> top_threshold = 30;
> min_storm_size = 16;
> max_storm_size = 1e+09;
> hail_dbz_threshold = 55;
> ZR = { 200.0, 1.6 };
> ZM = { 20465, 1.75 };
> min_radar_tops = 4.5;
> tops_edge_margin = 1.5;
> storm_data_dir = "$(DATA_DIR)/titan/storms/merged";
> create_verification_files = TRUE;
> verify_url = "mdv/verify";
> tracking_min_history_for_valid_forecast = 2400;
> tracking_min_fraction_overlap = 0.3;
```

O arquivo de parâmetros titan.merged fica no diretório $PROJ_DIR/titan/params e o script de inicialização start_Titan.merged em $PROJ_DIR/titan/scripts.

A aplicação Titan gera arquivos de em formato próprio (formato TITAN), que são criados um para cada dia e usam a hora como chave primária na recuperação do dado. Existem dois tipos de dados: dados das propriedades da tempestade (localização, volume, altura, etc) e dados da trajetória da tempestade (velocidade, previsão de intensificação, etc). Para cada tipo de dado existem dois arquivos, um para armazenar o dado e outro para armazenar o cabeçalho do dado. Os arquivos são nomeados como segue:

aaaammdd.sh5 (cabeçalho da tempestade)

aaaammdd.sd5 (dados da tempestade)

aaaammdd.th5 (cabeçalho da trajetória)

aaaammdd.td5 (dados da trajetória)

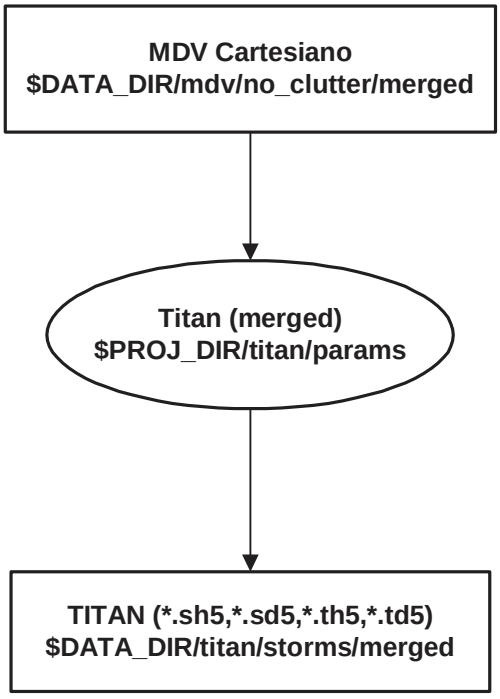
O programa Titan permite o cálculo de diversos parâmetros, que indicam a localização da tempestade, o volume, altura, área, a máxima refletividade e detalhes do rastreamento das tempestades. Outros parâmetros derivados também são calculados, incluindo, massa, fluxo de precipitação, métricas de granizo e de tempestades severas. Esses parâmetros podem ser analisados através dos visualizadores gráficos Rview e CIDD.

Com o objetivo de permitir análises através de ferramentas de uso geral como planilhas eletrônicas, banco de dados, pacotes estatísticos e ferramentas gráficas foram desenvolvidos programas utilitários que geram arquivos de saída ASCII facilmente importados por estas ferramentas. Alguns desses programas são:

- Mdv2Ascii;
- Tracks2Ascii

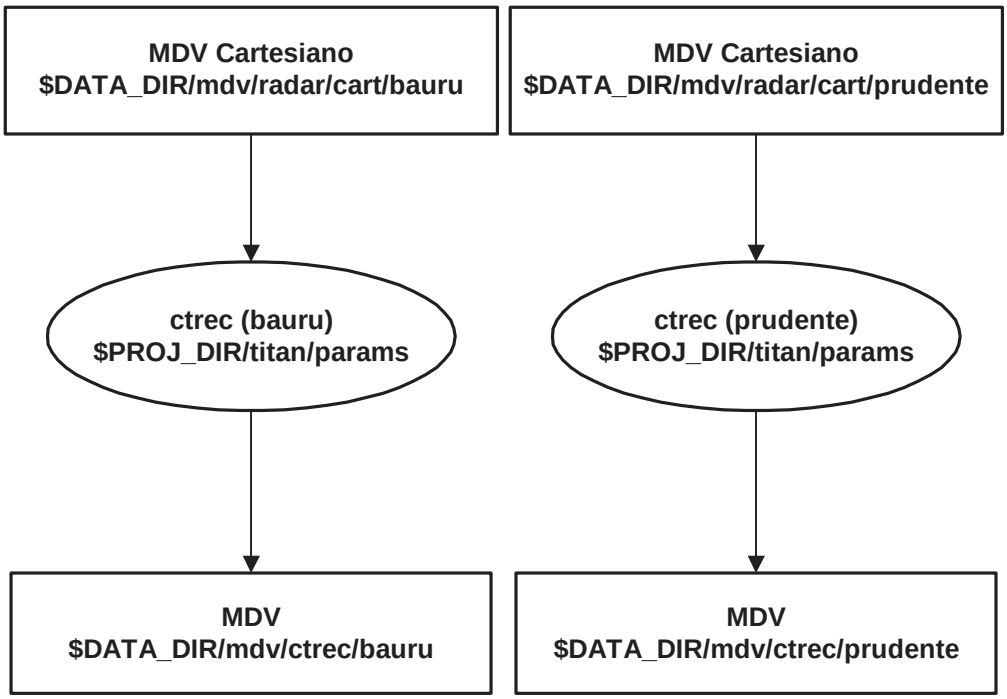
A aplicação VerifyTracks calcula estatísticas de verificação das previsões do Titan e imprime os dados na saída padrão (tela), além de atualizar os campos de verificação no arquivo de trajetória (“track”) da tempestade.

O fluxo de dados desta aplicação é apresentado a seguir:



O programa CTREC (Cartesian Tracking Radar Echoes by Correlation) realiza a previsão do deslocamento dos ecos de chuva usando a técnica de correlação cruzada para identificar padrões de comportamento entre duas chuvas em tempos consecutivos. Este programa é apropriado para realização do nowcasting de chuvas estratiformes. Os dados de entrada devem estar no formato MDV (refletividade) e a saída são campos de vetoriais (U/V) em formato MDV.

O fluxo de dados da aplicação CTREC é apresentado a seguir:




## 3.6 Aplicação PrecipAccum

O radar pode estimar a quantidade de precipitação no solo realizando uma integração no tempo da taxa de precipitação R (mm/h). A taxa de precipitação é calculada sobre o campo “Composite”. O TITAN trabalha dados sobre o volume completo, sendo que dados de diferentes altitudes podem ser vistos separadamente. Entretanto, às vezes é conveniente visualizar um sumário de padrões de refletividade sem precisar observar cada nível de altitude separadamente. O produto Composite foi criado com este propósito: ele observa a maior refletividade em cada coluna vertical dentro do domínio do sistema.

Várias instâncias do programa PrecipAccum são ativadas: uma que calcula a taxa de precipitação R a cada execução do radar (single), e outras para gerar a precipitação acumulada para o período de 1 hora e outra para o período de 24 horas.

Abaixo são apresentados alguns parâmetros do arquivo PrecipAccum.single:

```
< instance = "single";
< input_rdata_dir = "$(DATA_DIR)/mdv/no_clutter/merged";
< check_input_geom = FALSE;
< input_is_precip = FALSE;
< accum_method = SINGLE_FILE;
< adjust_for_expected_total_duration = FALSE;
< low_dbz_threshold = 0;
< data_set_source="Computed by applying ZR function to dBZ data and
integrating.";
< generate_rate_grid = FALSE;
< output_precip_dir = "$(DATA_DIR)/mdv/precip/single";
```

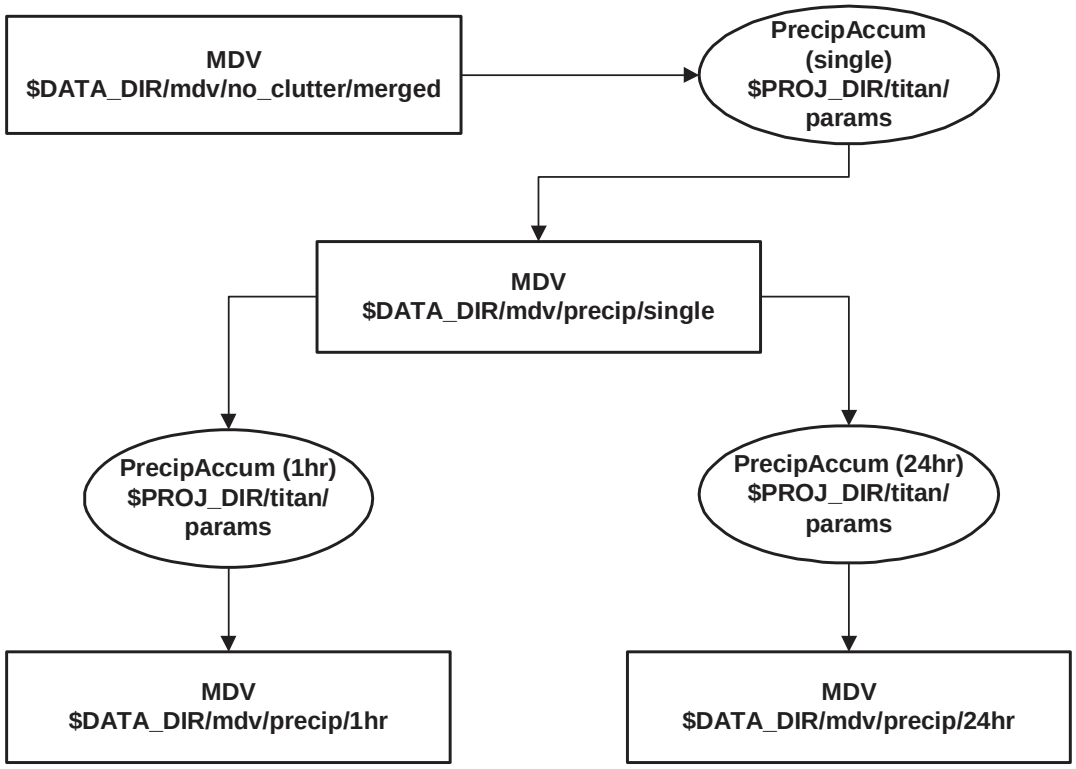
Segue abaixo alguns parâmetros do arquivo PrecipAccum.1hr:

```
> instance = "1hr";
> input_rdata_dir = "$(DATA_DIR)/mdv/precip/single";
> check_input_geom = TRUE;
> input_is_precip = TRUE;
> accum_method = RUNNING_ACCUM;
> adjust_for_expected_total_duration = TRUE;
> low_dbz_threshold = 10;
> data_set_source = "Computed by summing single-file precip values.";
> generate_rate_grid = TRUE;
> output_precip_dir = "$(DATA_DIR)/mdv/precip/1hr"
```

A seguir alguns parâmetros do arquivo PrecipAccum.24hr:

```
> instance = "24hr";
> input_rdata_dir = "$(DATA_DIR)/mdv/precip/single";
> check_input_geom = TRUE;
> input_is_precip = TRUE;
> accum_method = ACCUM_FROM_TIME_OF_DAY;
> adjust_for_expected_total_duration = TRUE;
> low_dbz_threshold = 10;
> data_set_source = "Computed by summing single-file precip values.";
> generate_rate_grid = TRUE;
> output_precip_dir = "$(DATA_DIR)/mdv/precip/24hr";
```

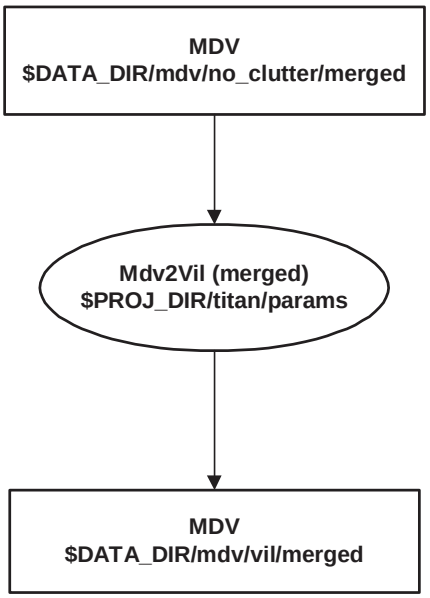
A seguir o fluxo de dados desta aplicação:



## 3.7 Aplicação Mdv2Vil

A aplicação Mdv2Vil lê o volume de radar em formato MDV e calcula uma estimativa do conteúdo líquido de água em cada coluna vertical da área coberta pelo radar. O cálculo do VIL é realizado utilizando-se a relação Z-m, onde m é a densidade de vapor em $g/m^3$, e integrando sobre a coluna: $Z=20465*m^{1.75}$. Os campos DVIL (diferença entre o VIL acima de um certo nível e o VIL abaixo deste nível) e o índice SSS (Storm Severity Structure) também são calculados e incluídos no arquivo de saída.

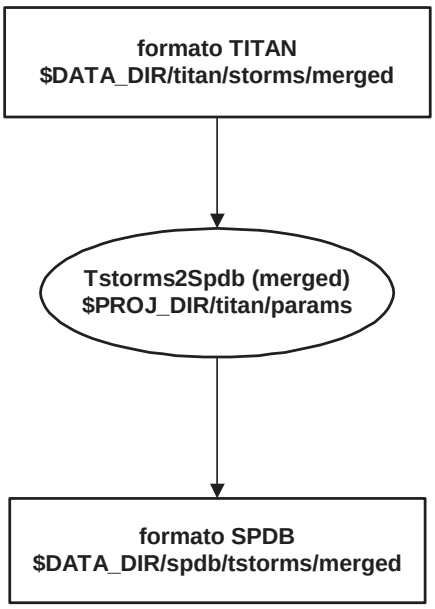
O fluxo de dados abaixo exemplifica a execução da aplicação Mdv2Vil:



## 3.8 Aplicação Tstorm2Spdb

Este programa lê os arquivos de tempestades identificadas e de previsão de trajetórias das mesmas, gerados pela aplicação Titan e converte as propriedades do rastreamento no formato SPDB.

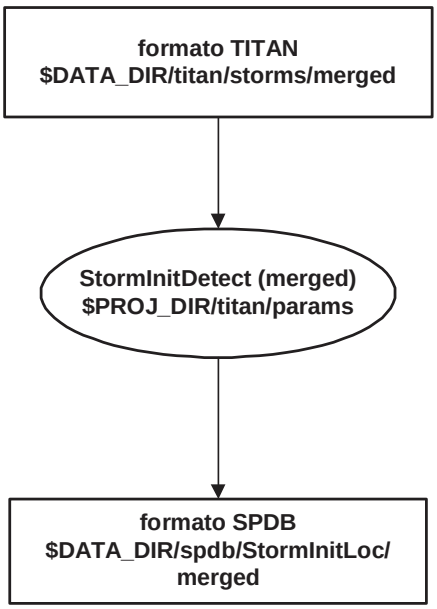
Abaixo uma descrição do fluxo de dados desta aplicação:



## 3.9 Aplicação StormInitDetect

Esta aplicação lê o dado resultante da execução do programa Titan, determina quais tempestades são consideradas significantes de acordo com alguns critérios e grava um arquivo de latitudes e longitudes da posição inicial dessas tempestades no formato SPDB.

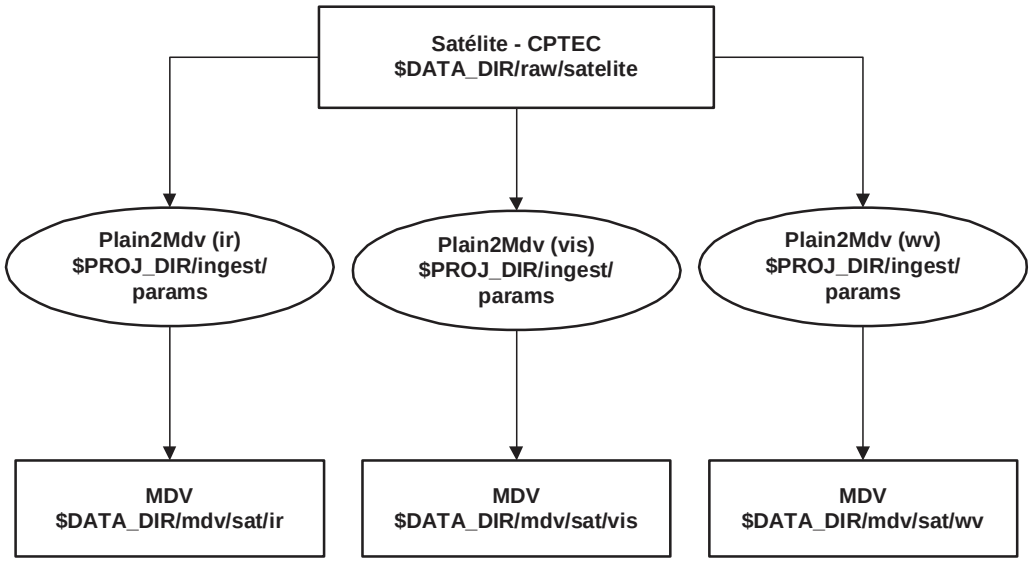
Abaixo o fluxo de dados da aplicação StormInitDetect:



## 3.10 Introdução de Dados de Satélite

A introdução de dados do satélite Goes provindo do CPTEC-INPE é realizada pela execução do programa Plain2Mdv. Os arquivos de configuração Plain2Mdv.ir, Plain2Mdv.vis e Plain2Mdv.wv são criados para gerar instâncias diferentes do programa Plain2Mdv na criação dos arquivos de satélite no formato infravermelho, visível e vapor d'água respectivamente.

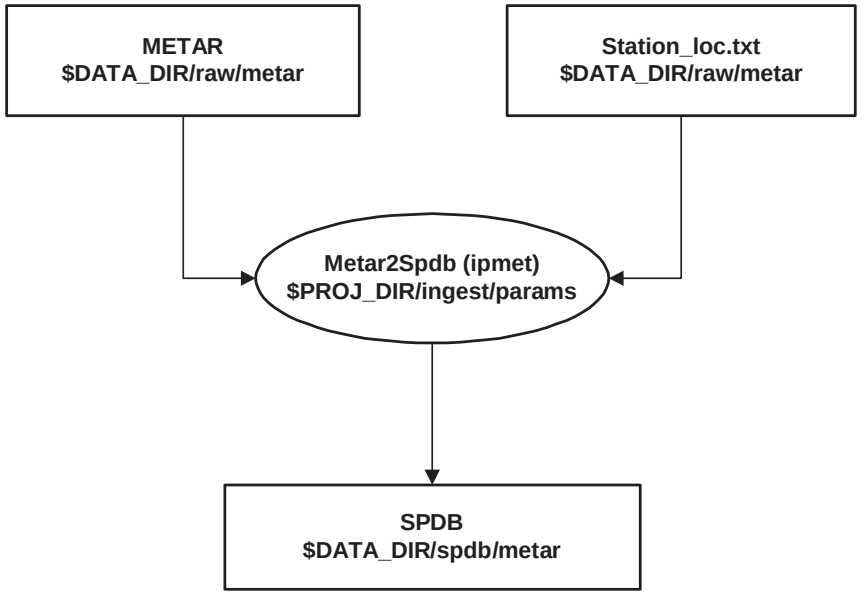
A seguir o fluxo de dados da aplicação Plain2Mdv:



## 3.11 Introdução de dados de METAR

A introdução de dados de METAR é realizada pelo programa Metar2Spdb, e requer também a definição do arquivo com a localização das estações de METAR (Station_loc.txt). Dados de METAR são armazenados no formato SPDB.

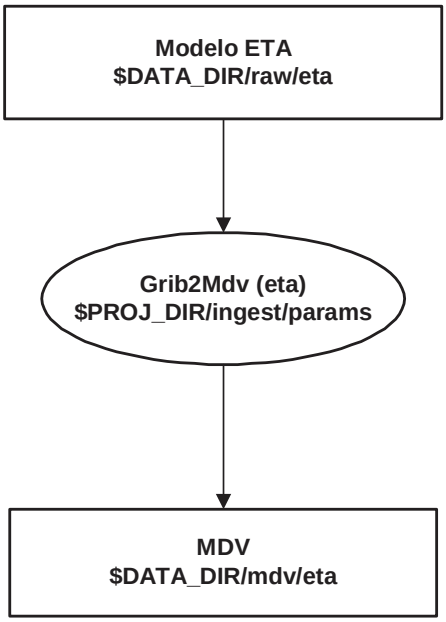
Abaixo o fluxo de dados da aplicação Metar2Spdb:



## 3.12 Introdução do Modelo Numérico de Previsão ETA

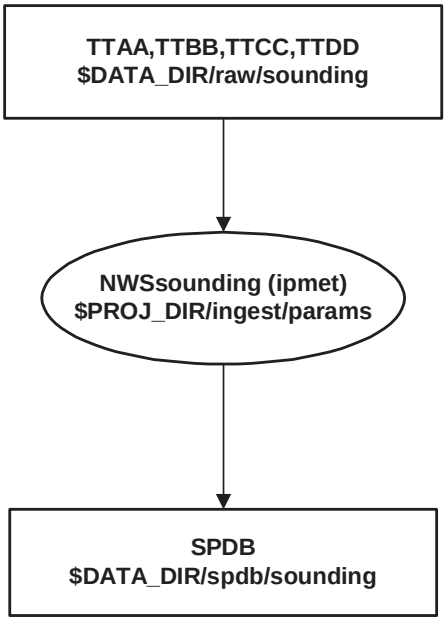
O modelo numérico de previsão do tempo ETA executado no IPMet é gerado no formato GRIB, e este formato é convertido para o MDV através do programa Grib2Mdv. No arquivo de configuração desta aplicação devem ser definidos os campos que se deseja gerar no formato MDV.

O fluxo de dados da aplicação pode ser observado abaixo:



## 3.13 Introdução de Dados de Radiossondas

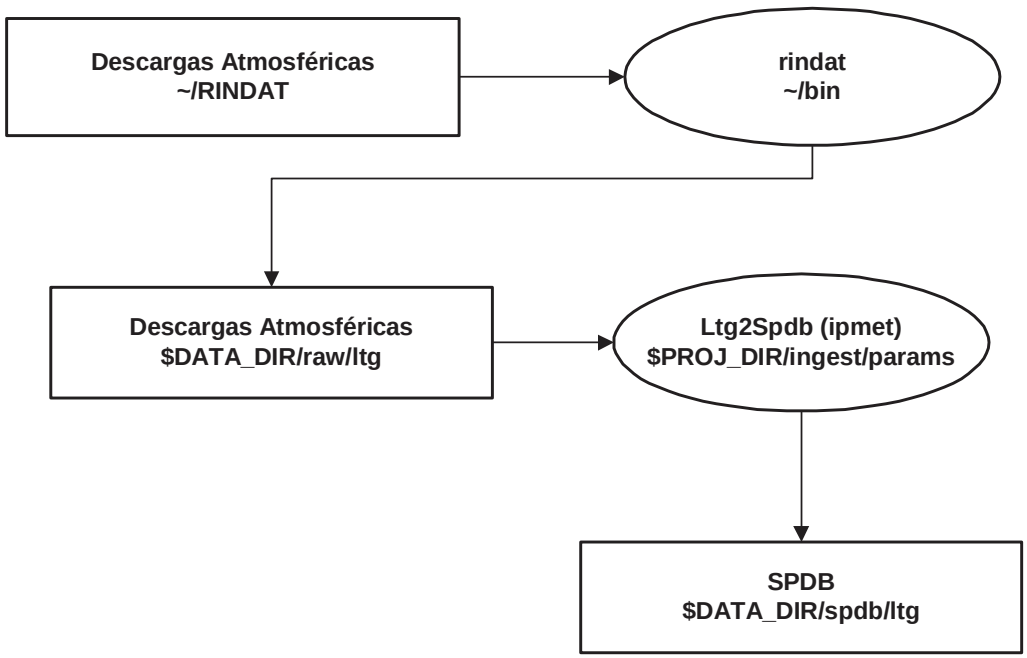
Os dados de estações de radiossondagens são recebidos no IPMet no formato TTAA, TTBB, TTCC e TTDD. A aplicação NWSsounding2Spdb é responsável pela conversão deste formato no formato SPDB. Ela requer também o arquivo de latitudes e longitudes das estações de sondagens.



## 3.14 Introdução de Dados de Descargas Atmosféricas

Os dados originais de descargas atmosféricas (RINDAT) devem ser primeiramente cornvertidos para o formato ASCII compreendido pelo programa Ltg2Spdb. O programa rindat.c foi desenvolvido para esta finalidade, e coloca o arquivo de saída gerado no diretório $DATA_DIR/raw/ltg.

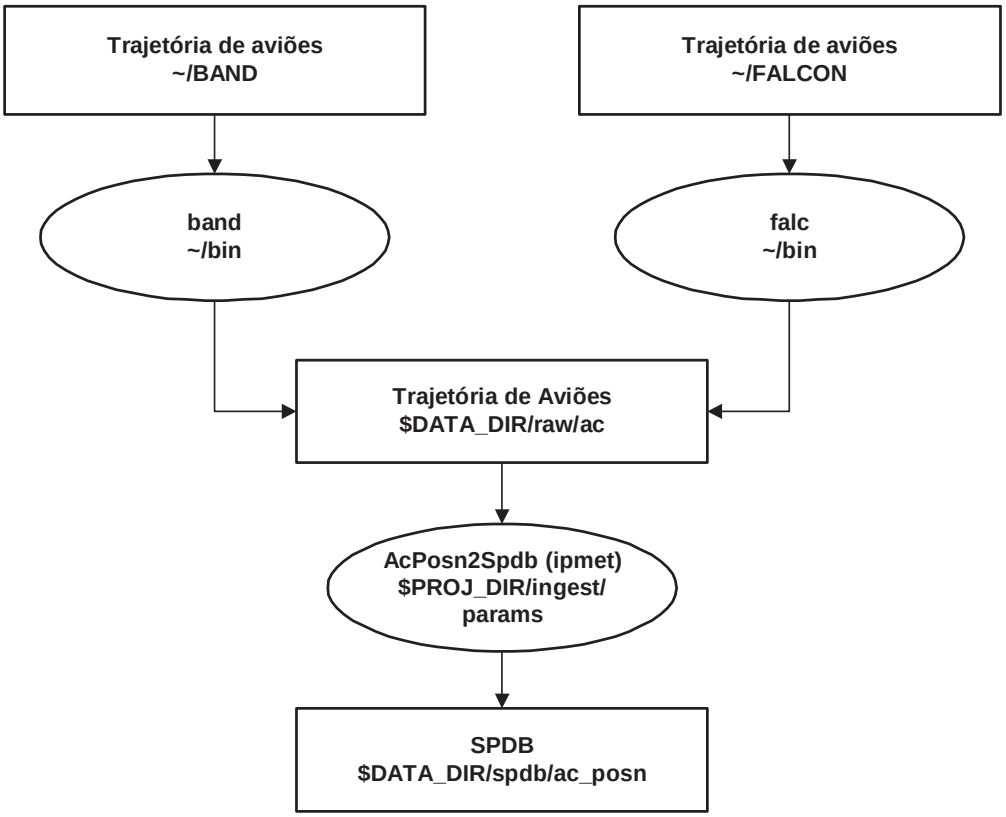
O programa Ltg2Spdb é responsável pela conversão dos dados de descargas atmosféricas para o formato interno do TITAN:



## 3.15 Introdução de Dados de Trajetórias de Aviões

Os dados de trajetória dos aviões Bandeirantes e Falcon em seu formato original devem passar por programas conversores de formato, para que atendam as especificidades do programa AcPosn2Spdb. Foram desenvolvidos os programas Band.c e Falc.c para esta finalidade. Os dados ASCII gerados ficam em $DATA_DIR/raw/ac.

O programa AcPosn2Spdb realiza a conversão do formato em $DATA_DIR/raw/ac para o formato SPDB:



## 3.16 Aplicações Gráficas de Visualização (Rview e CIDD)

### Rview

Aplicação para visualização de dados e produtos do sistema TITAN, que é capaz de ler os dados no formato TITAN original, fornecendo inúmeras funcionalidades para identificação de tempestades e análise da trajetória das mesmas. Funciona juntamente com o programa TimeHist, que apresenta o histórico das propriedades da tempestade no tempo.

Os programas Rview e TimeHist utilizam arquivos de parâmetros para configuração do ambiente de visualização. Alguns parâmetros da aplicação Rview podem ser observados a seguir:

```
> Rview.instance: merged
> Rview.scan_delta_t: 600
> Rview.z_requested: 2.0
> Rview.projection: latlon
> Rview.grid_lat: -22.0
> Rview.grid_lon: -50.0
> Rview.full_min_x: -54.0
> Rview.full_min_y: -24.5
> Rview.full_max_x: -46.5
> Rview.full_max_y: -19.5
> Rview.plot_rings: false
> Rview.map_conf_file: $(PROJ_DIR)/display/maps/merged.conf
> Rview.cursor_magnetic: false
> Rview.time_hist_command_line: TimeHist -params $(PROJ_DIR)/display/params/TimeHist.merged &
> Rview.titan_server_url: titanp:://localhost::titan/storms/merged
> Rview.track_shmem_key: 46300
> Rview.plot_ac_posn: false
> #fc dBZ mdvp:://localhost::mdv/no_clutter/merged DBZ 720 dbz_color dbz_gray 0.0 100.0 5.0
> #fc Vel mdvp:://localhost::mdv/no_clutter/merged VEL 720 vel_color vel_gray 0.0 100.0 5.0
> #fc 1hr_Precip_Accum mdvp:://localhost::mdv/precip/1hr precip 720 precip_color precip_gray 0.0 100.0 5.0
> #fc 24hr_Precip_Accum  mdvp:://localhost::mdv/precip/24hr precip 720 precip_color precip_gray 0.0 100.0 5.0
> #fc 1hr_Precip_Rate mdvp:://localhost::mdv/precip/1hr rate 720 precip_color precip_gray 0.0 100.0 5.0
> #fc Vil mdvp:://localhost::mdv/vil/merged vil 720 vil_color vil_gray 0.0 100.0 5.0
> #fc Vil_Overhang mdvp:://localhost::mdv/vil/merged dVil 720 vil_diff_color vil_diff_gray 0.0 100.0 5.0
> #fc SSS mdvp:://localhost::mdv/vil/merged SSS 720 sss_color sss_gray 0.0 100.0 5.0
```

Os arquivos de configuração dos programas Rview e TimeHist no projeto IPMet estão localizados em $PROJ_DIR/display/params.

O programa para inicialização e finalização dos programas Rview/TimeHist estão em $PROJ_DIR/display/scripts: start_Rview.merged e kill_Rview.merged. Eles devem ser acionados pelo programa procmap e devem constar no arquivo $PROJ_DIR/control/proc_list para execução automática em tempo real.

Para executar manualmente, digitar: `Rview -params Rview.merged`.

## CIDD (Cartesian Interactive Data Display)

Aplicação de visualização avançada para o TITAN, que apresenta dados em formato MDV e SPDB. Possibilita a visualização de vários tipos de dados meteorológicos, como satélite, modelos numéricos, metares, descargas elétricas, trajetória de aviões, entre outros.

O programa CIDD utiliza o arquivo de parâmetros para realizar a configuração dos menus e produtos visualizados. Ele está subdividido em algumas seções que são delimitadas por <NOME_SEÇÃO> E </NOME_SEÇÃO>.

As seções atualmente suportadas são:

GRIDS – descreve as fontes de dados MDV e características de visualização;

WINDS – descreve as fontes de dados para apresentação como vetores;

MAPS – descreve os arquivos de overlay;

MAIN_PARAMS – controla as principais funções da visualização;

DRAW_EXPORT – descreve produtos para serem manualmente desenhados;

SYMPRODS – descreve os produtos no formato SPDB;

TERRAIN – descreve arquivos de terreno;

ROUTE_WINDS – descreve características pré-definidas de rotas.

O arquivo de parâmetros de configuração encontra-se em $PROJ_DIR/display/params e o script de inicialização está em $PROJ_DIR/display/scripts. Para execução manual, digitar CIDD –p CIDD.merged, no diretório $PROJ_DIR/display/params.

Abaixo encontra-se uma amostra das seções GRIDS, WINDS e SYMPRODS do arquivo de configuração CIDD.merged do projeto IPMet:

```
> <GRIDS>
> #
> # Radar Data
> #
> ###################################
> #
> # Merged.
> #
> # Cartesian. Composite first.
> #
> Merged_Comp_Cart_ref Merged_Composite_Cart-dbz   mdvp:://localhost::mdv/no_clutter/merged&DBZ dbz.colors      dBZ     0 2 1
comp     1 0
> #
> Merged_Cart_ref Merged_Cart-dbz   mdvp:://localhost::mdv/no_clutter/merged&DBZdbz.colors     dBZ     0 2 1 dcont     1 0
> Merged_Cart_vel Merged_Cart-vel   mdvp:://localhost::mdv/no_clutter/merged&VELbauruVel.colors m/sec   0 2 1 dcont     1 0
> #
```

> # clutter removed
> #
> NoClutterBruDbz NoClutterBruDbz mdvp::://localhost::mdv/no_clutter/bauru&DBZ dbz.colors dBZ 0 2 1 dcont 1 0
> NoClutterBruVel NoClutterBruVel mdvp::://localhost::mdv/no_clutter/bauru&VEL bauruVel.colors m/s 0 2 1 dcont 1 0
> NoClutterPprDbz NoClutterPprDbz mdvp::://localhost::mdv/no_clutter/prudente&DBZ dbz.colors dBZ 0 2 1 dcont 1 0
> NoClutterPprVel NoClutterPprVel mdvp::://localhost::mdv/no_clutter/prudente&VEL bauruVel.colors m/s 0 2 1 dcont 1 0
> ##################################
> #
> # Bauru.
> #
> # Cartesian. Composite first.
> #
> Bauru_Comp_Cart_ref Bauru_Composite_Cart-dbz mdvp::://localhost::mdv/radar/cart/bauru&DBZ dbz.colors dBZ 0 2 1 comp 1 0
> #
> Bauru_Cart_ref Bauru_Cart-dbz mdvp::://localhost::mdv/radar/cart/bauru&DBZ dbz.colors dBZ 0 2 1 dcont 1 0
> Bauru_Cart_vel Bauru_Cart-vel mdvp::://localhost::mdv/radar/cart/bauru&VEL bauruVel.colors m/sec 0 2 1 dcont 1 0
> #
> # Polar.
> Bauru_polar_ref Bauru_polar-dbz mdvp::://localhost::mdv/radar/polar/bauru&DBZdbz.colors dBZ 0 2 1 dcont 1 0
> Bauru_polar_vel Bauru_polar-vel mdvp::://localhost::mdv/radar/polar/bauru&VELbauruVel.colors m/sec 0 2 1 dcont 1 0
> ##############################################
> # Prudente.
> # Cartesian. Composite first.
> Prudente_Comp_Cart_ref Prudente_Composite_Cart-dbz mdvp::://localhost::mdv/radar/cart/prudente&DBZ dbz.colors dBZ 0 2 1 comp 1 0
> #
> Prudente_Cart_ref Prudente_Cart-dbz mdvp::://localhost::mdv/radar/cart/prudente&DBZ dbz.colors dBZ 0 2 1 dcont 1 0
> Prudente_Cart_vel Prudente_Cart-vel mdvp::://localhost::mdv/radar/cart/prudente&VEL bauruVel.colors m/sec 0 2 1 dcont 1 0
> #
> # Polar.
> #
> Prudente_polar_ref Prudente_polar-dbz mdvp::://localhost::mdv/radar/polar/prudente&DBZ dbz.colors dBZ 0 2 1 dcont 1 0
> Prudente_polar_vel Prudente_polar-vel mdvp::://localhost::mdv/radar/polar/prudente&VEL bauruVel.colors m/sec 0 2 1 dcont 1 0
>
> # VIL
>
> Merged_VIL Merged_VIL mdvp::://localhost::mdv/vil/merged&vil vil_color kg/m2 0 2 1 dcont 1 0
> Merged_DVIL Merged_DVIL mdvp::://localhost::mdv/vil/merged&dVil vil_diff_color kg/m2 0 2 1 dcont 1 0
> Merged_SSS Merged_SSS mdvp::://localhost::mdv/vil/merged&SSS sss_color index 0 2 1 dcont 1 0
>
> # PRECIP
>
> PRECIP_1hr PRECIP_1hr mdvp::://localhost::mdv/precip/1hr&precip precip_color mm 0 2 1 dcont 1 0
> PRECIP_rate PRECIP_rate mdvp::://localhost::mdv/precip/1hr&rate precip_color mm 0 2 1 dcont 1 0
> PRECIP_24hr PRECIP_24hr mdvp::://localhost::mdv/precip/24hr&precip precip_color mm 0 2 1 dcont 1 0
> #PRECIP_24hr PRECIP_24hr mdvp::://localhost::mdv/precip/24hr&precip precip_color mm 0 2 1 dcont 0 0
>
> # ClutterMap
> ClutterMapBru ClutterMapBru mdvp::://localhost::mdv/clutter/bauru&DBZMed dbz.colors dBZ 0 2 1 comp 1 0
> ClutterMapPpr ClutterMapPpr mdvp::://localhost::mdv/clutter/prudente&DBZMed dbz.colors dBZ 0 2 1 comp 1 0
> # Terrain
> #
> Terrain Terrain mdvp::static//localhost::mdv/terrain&Elevation terrain.colors m 0 2 1 cart 1 0
> </GRIDS>
>##################################################################################################################
><WINDS>
>TREC_bauru mdvp::://localhost::mdv/ctrec/bauru& U^comp V^comp None m/s 1 white
>TREC_prudente mdvp::://localhost::mdv/ctrec/prudente& U^comp V^comp None m/s 1 white
</WINDS>

```
>#########################################################################################################
><SYMPRODS>
>  prod_info = {
>  {
>    menu_label = "TITAN StormsDetect",
>    url = "spdbp:Tstorms2Symprod:titanDetect//localhost::spdb/tstorms/merged",
>    data_type = 0,
>    render_type = RENDER_LATEST_IN_FRAME,
>    on_by_default = TRUE,
>    minutes_allow_before = 0.0,
>    minutes_allow_after = 0.0,
>    text_off_threshold = 0.4
>  },
>  {
>    menu_label = "TITAN StormsExtrap30",
>    url = "spdbp:Tstorms2Symprod:titanExtrap30//localhost::spdb/tstorms/merged",
>    data_type = 0,
>    render_type = RENDER_LATEST_IN_FRAME,
>    on_by_default = TRUE,
>    minutes_allow_before = 0.0,
>    minutes_allow_after = 0.0,
>    text_off_threshold = 0.4
>  },
>  {
>    menu_label = "TITAN StormsVectors",
>    url = "spdbp:Tstorms2Symprod:titanVectors//localhost::spdb/tstorms/merged",
>    data_type = 0,
>    render_type = RENDER_LATEST_IN_FRAME,
>    on_by_default = TRUE,
>    minutes_allow_before = 0.0,
>    minutes_allow_after = 0.0,
>    text_off_threshold = 0.4
>  },
>  {
>     menu_label = "Storm Init Locations",
>     url = "spdbp:GenPt2Symprod:init//localhost::spdb/StormInitLoc/merged",
>     data_type = 0,
>     render_type = RENDER_LATEST_IN_FRAME,
>     on_by_default = FALSE,
>     minutes_allow_before = 60.0,
>     minutes_allow_after = 0.0,
>     text_off_threshold = 0
>   },
>   {
>     menu_label = "METARS - simple",
>     url = "spdbp:Metar2Symprod:simple//localhost::spdb/metar",
>     data_type = 0,
>     render_type = RENDER_LATEST_IN_FRAME,
>     on_by_default = TRUE,
>     minutes_allow_before = 60.0,
>     minutes_allow_after = 0.0,
>     text_off_threshold = 0.4
>   },
>   {
>     menu_label = "METARS - with labels",
>     url = "spdbp:Metar2Symprod:labels//localhost::spdb/metar",
>     data_type = 0,
>     render_type = RENDER_LATEST_IN_FRAME,
>     on_by_default = TRUE,
>     minutes_allow_before = 60.0,
>     minutes_allow_after = 0.0,
>     text_off_threshold = 0.4
```

```
>  },
>  {
>    menu_label = "Aircraft Tracks",
>    url = "spdbp:_PosnRpt2Symprod:init//localhost::spdb/ac_posn",
>    data_type = 0,
>    render_type = RENDER_LATEST_IN_FRAME,
>    on_by_default = TRUE,
>    minutes_allow_before = 0.0,
>    minutes_allow_after = 0.0,
>    text_off_threshold = 0
>  },
>  {
>     menu_label = "Ltg",
>     url = "spdbp:Ltg2Symprod:params//localhost::spdb/ltg",
>     data_type = 0,
>     render_type = RENDER_ALL_VALID,
>     on_by_default = TRUE,
>     minutes_allow_before = 60.0,
>     minutes_allow_after = 0.0,
>     text_off_threshold = 0
>  }
> };
> </SYMPRODS>
```

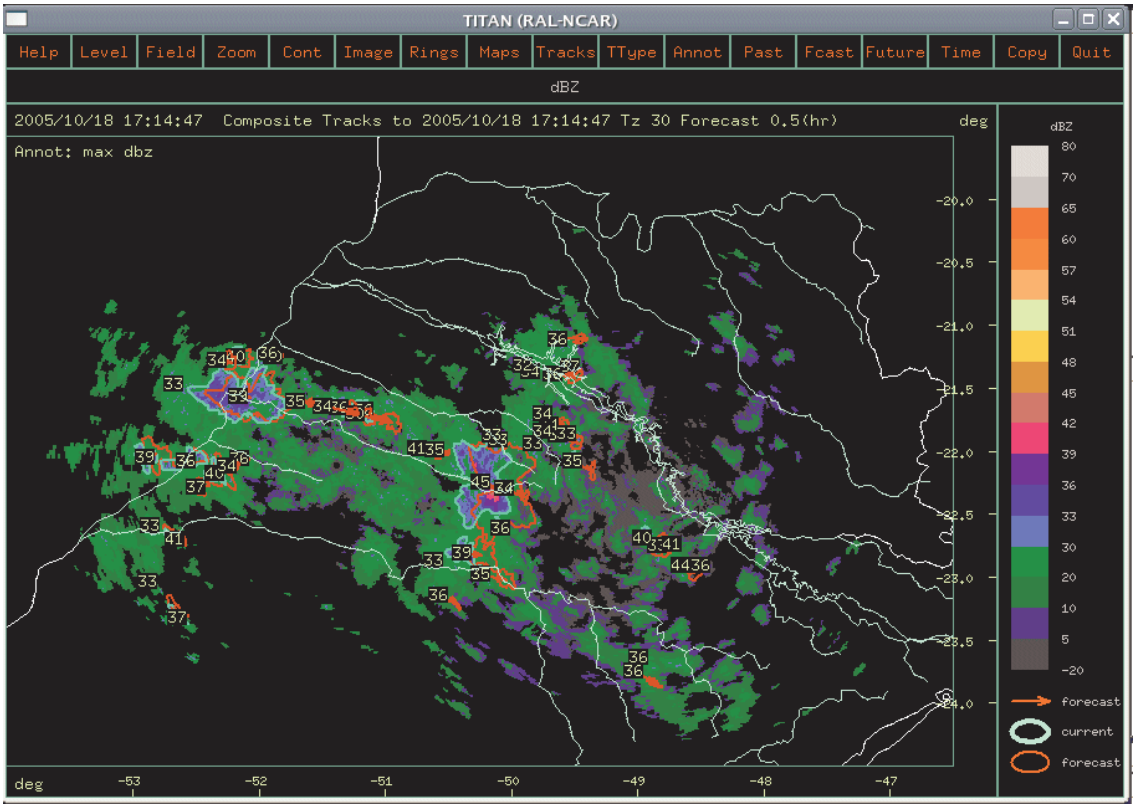
Uma observação em relação ao campo “url” da seção SYMPRODS. No exemplo:

menu_label = "TITAN StormsDetect",
url = "spdbp: Tstorms2Symprod: titanDetect //localhost:: spdb/tstorms/merged ",

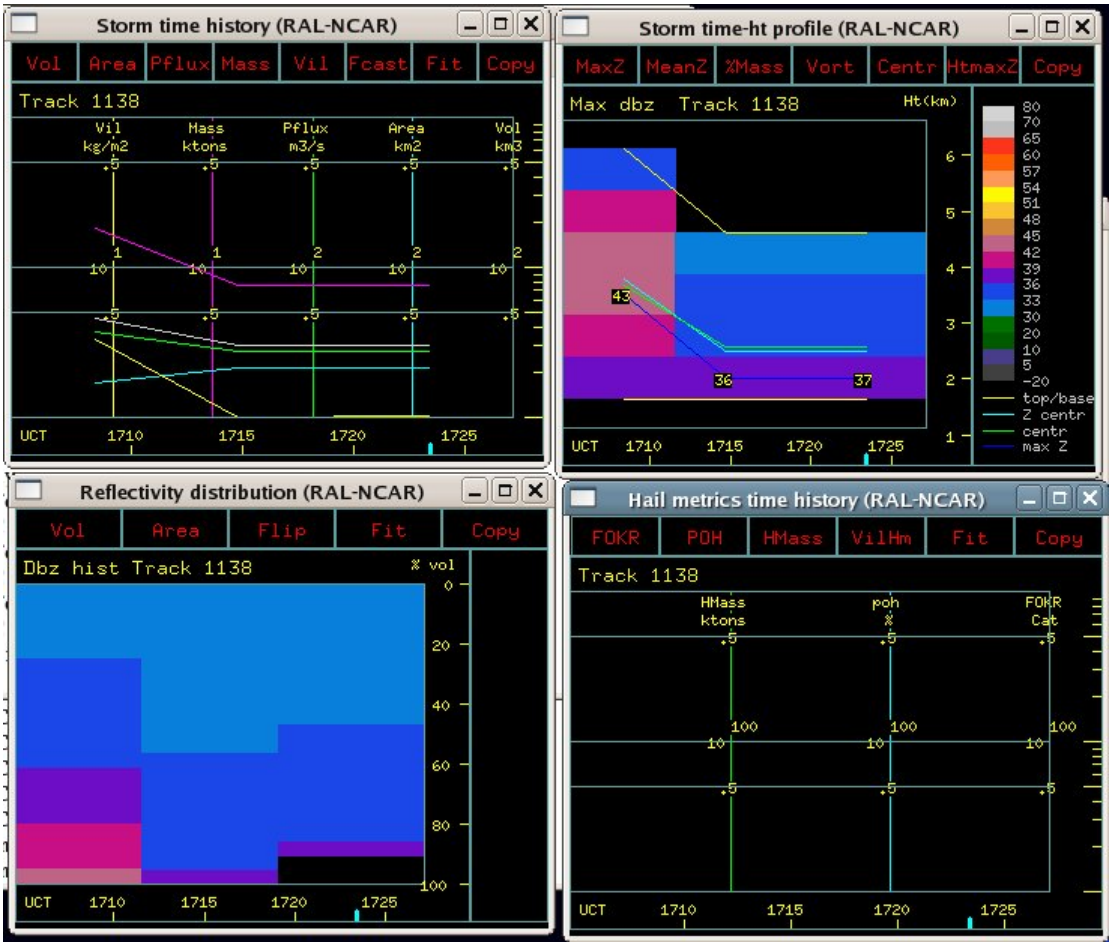
Tstorms2Symprod especifica o nome do arquivo de configuração do produto cujo rótulo no CIDD aparece como “TITAN StormsDetect” e titanDetect é a instância. O arquivo de configuração desse produto, cujo nome é _Tstorms2Symprod.titanDetect fica em $(DATA_DIR)/spdb/tstorms/merged. Portanto, quando for necessário fazer ajustes na apresentação gráfica do produto “TITAN StormsDetect" (cores, espessura de linha, etc), é necessário alterar o arquivo _Tstorms2Symprod.titanDetect. Pelo fato desse arquivo ficar armazenado na área de dados do sistema, é necessário incluir o caracter “_“ (*underscore*) antes do nome do arquivo para que ele não seja removido pelo processo Janitor (realiza a limpeza de dados do sistema).

A seguir são apresentadas algumas telas dos visualizadores Rview/TimeHist e CIDD:

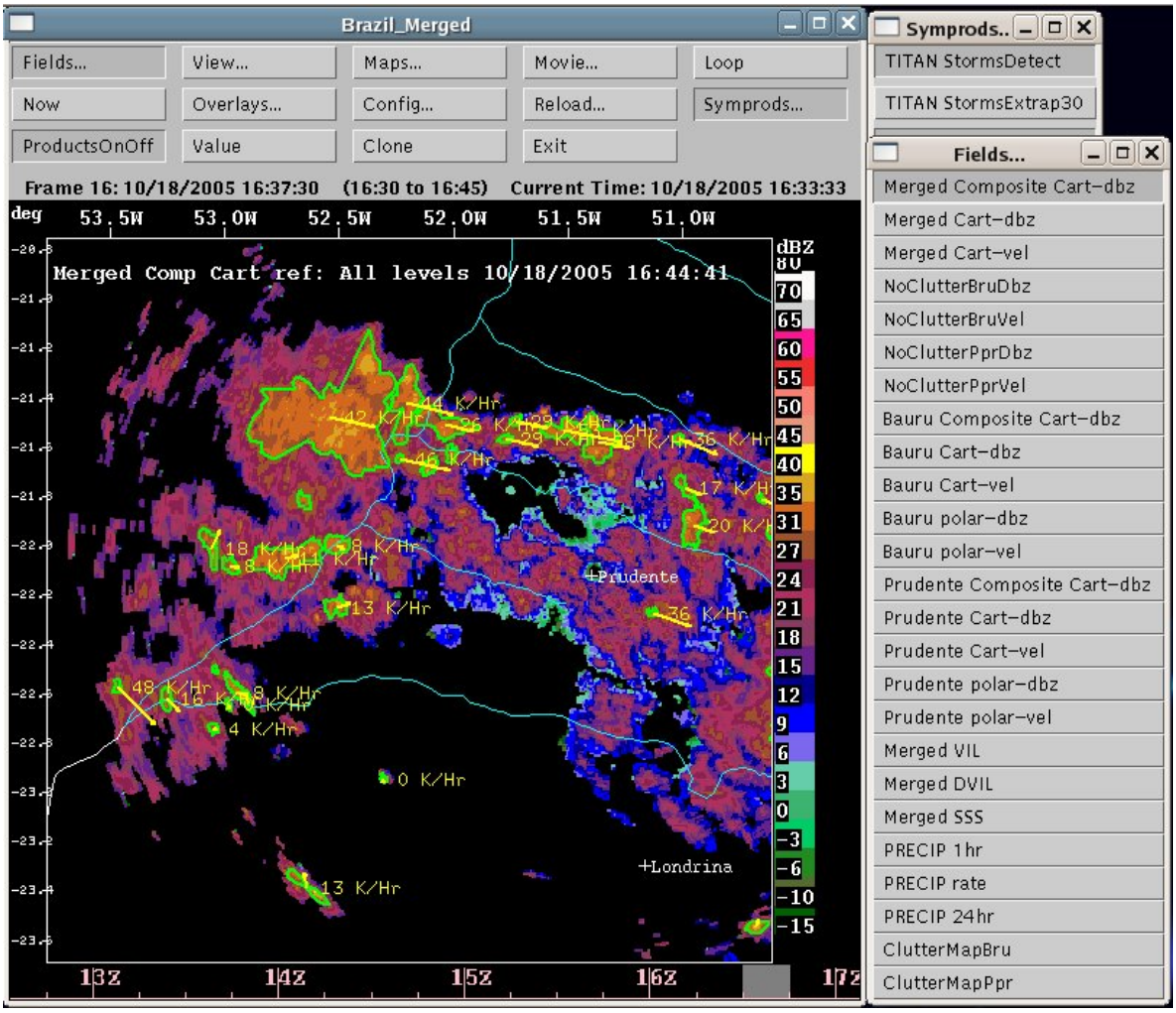
A figura abaixo mostra a tela do programa Rview com o produto Composite dos dois radares: Bauru e Presidente Prudente. No topo da aplicação encontra-se o menu de funções da aplicação e que fazem uso dos três botões do mouse. A funcionalidade desses botões é explicada numa janela após a escolha da função help e seleção de opções do menu de funções.



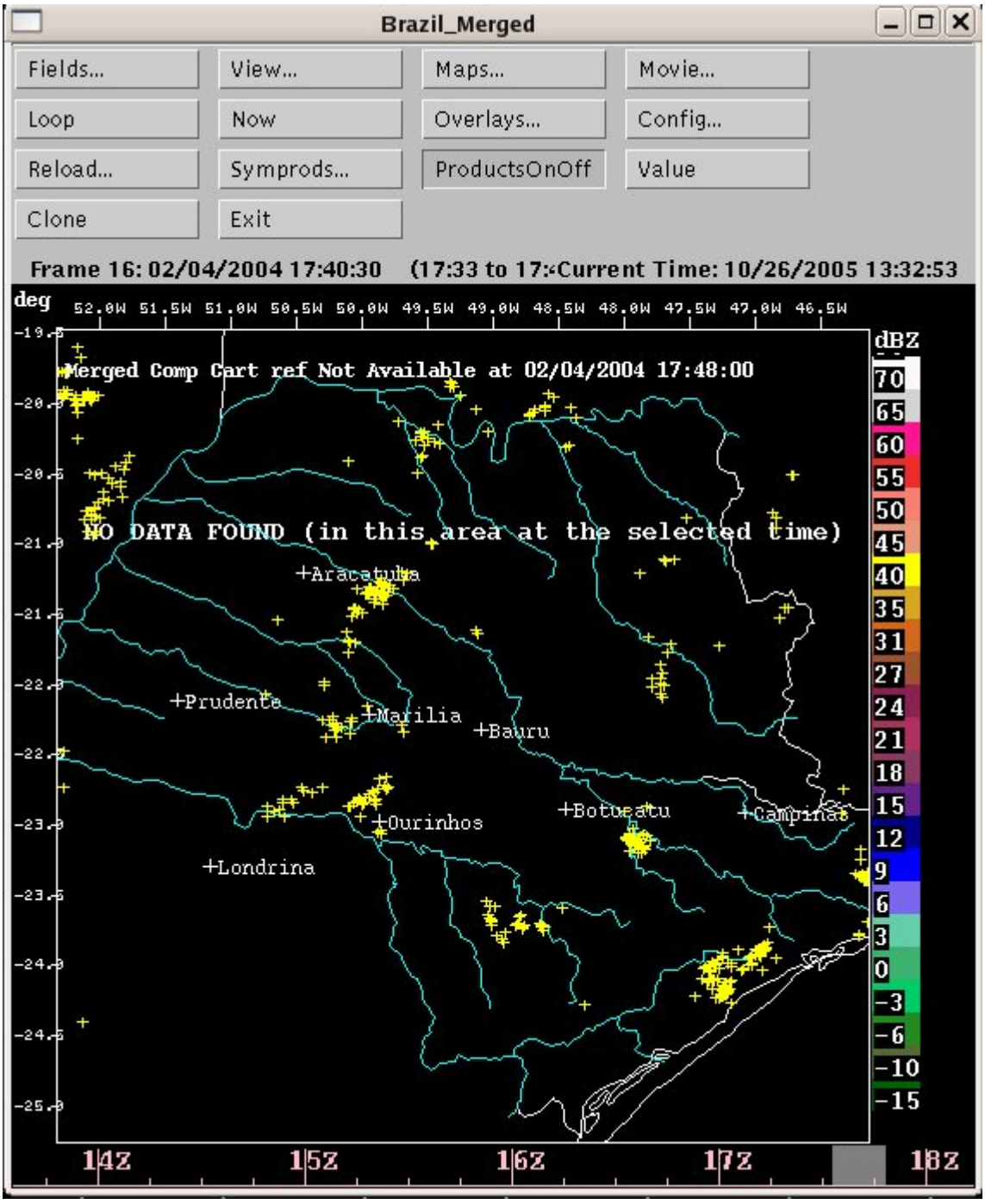
A seguir são apresentados alguns menus referentes à aplicação TimeHist, que permite a análise de diversas propriedades da tempestade no tempo:



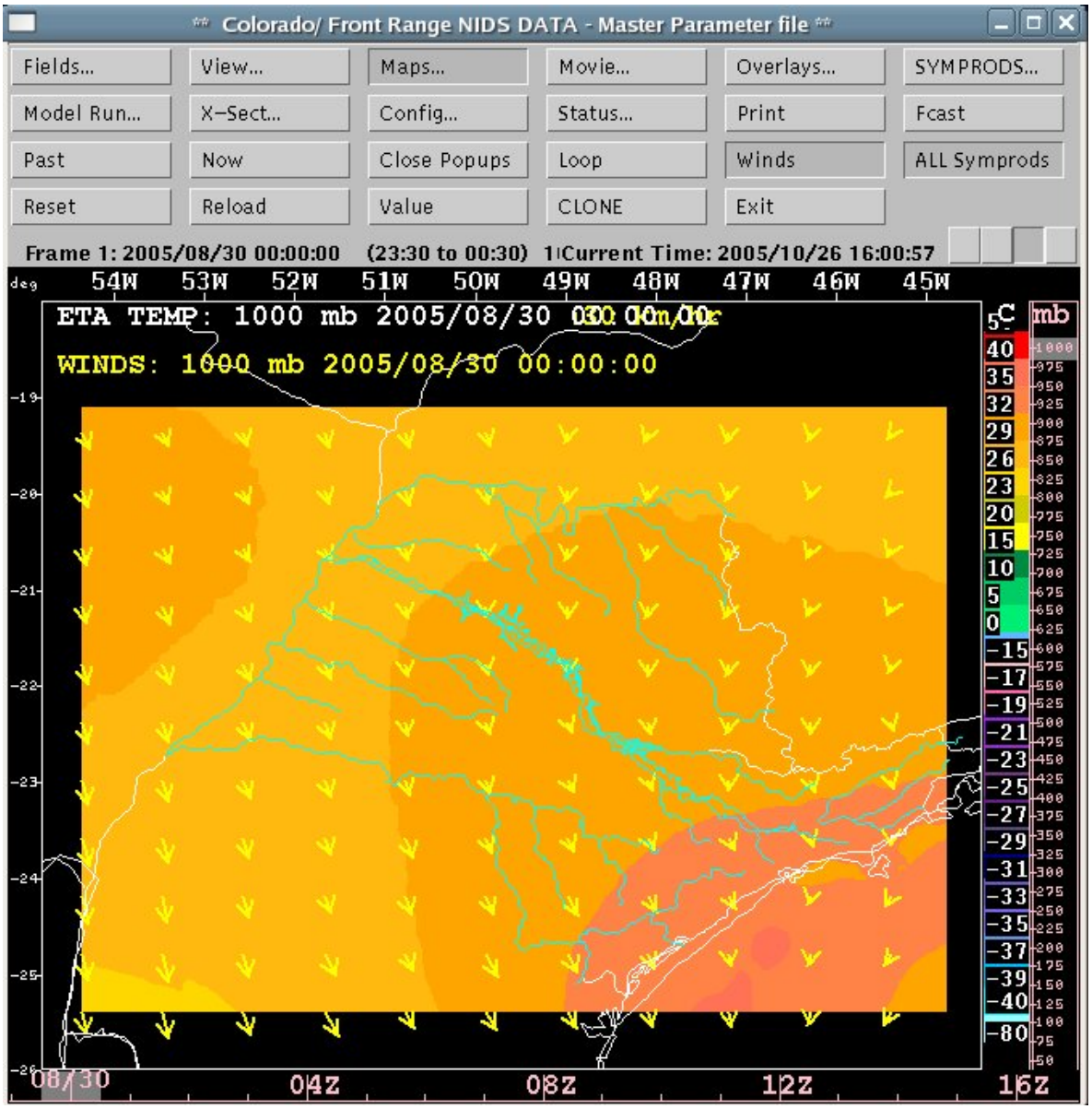
A seguir é apresentada a tela do programa CIDD mostrando o produto Composite com a fusão dos radares de Bauru e Presidente Prudente. Ao lado encontra-se o menu de produtos em formato MDV (Fields) e o menu de produtos em formato SPDB (Symprods):



A seguir é mostrado o programa CIDD apresentando o produto Descargas Atmosféricas:



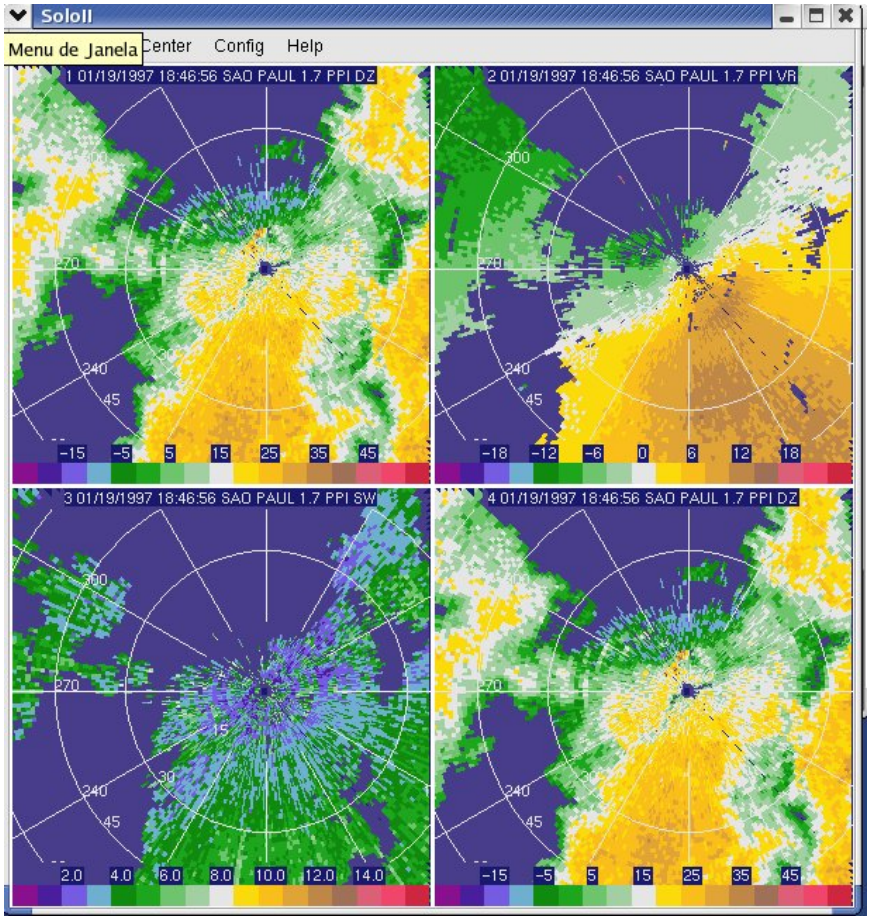
Na figura abaixo o programa CIDD apresenta o campo de vetores de vento sobreposto ao campo de temperatura do modelo ETA:



# 4. OUTRAS ATIVIDADES REALIZADAS

A visita ao National Center for Atmospheric Research (NCAR) possibilitou o contato com cientistas e engenheiros de software de outros laboratórios (não somente o RAL), tornado possível o conhecimento de outros sistemas que estão sendo desenvolvidos no NCAR.

O sistema SOLO está sendo desenvolvido pelo Earth Observing Laboratory e provê facilidades na análise de dados de radar. Abaixo é mostrado o programa SOLO com dados do radar do IPMet-Bauru:




O contato com a cientista Juanzhen Sun possibilitou o conhecimento sobre o software VDRAS (Variational Doppler Radar Analysis). Este sistema utiliza dados de um radar Doppler e técnicas de assimilação usando o método “adjoint” para recuperar campos tridimensionais de vento, termodinâmica e variáveis de nuvens. Ele está sendo usado para compor o sistema Auto-Nowcasting do NCAR. Referências sobre este sistema podem ser encontradas em Sun e Crook (1997,1998,2001).

## REFERÊNCIAS BIBLIOGRÁFICAS


COLLIER, C.G. **Short-period forecasting using radar data**, IN: Collier,C.G. Applications of Weather Radar Systems, 2.ed, 1996, England; **Praxis Publishing Ltd**, p. 165-224.

DIXON, M e WIENER G. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting. **Journal of Atmospheric and Oceanic Technology,** 1993, v.10 (3), p.785-797.

DIXON, M: **Automated Storm Identification, Tracking and Forecasting – A Radar-Based Method**, Phd Thesis no. 148, University of Colorado and National Center for Atmospheric Research, 1994.

DIXON, M: **TITAN User's Guide**, 2005, [ Disponível em www.rap.ucar.edu/projects/titan]

SUN J., N A CROOK. Dynamical and Microphysical Retrieval from Doppler Radar Observations Using a Cloud Model and Its Adjoint. Part I: Model Development and Simulated Data Experiments. **Bulletin of the American Meteorological Society,** 1997, v. 54, p. 1642-1661.

SUN J., N A CROOK. Dynamical and Microphysical Retrieval from Doppler Radar Observations Using a Cloud Model and Its Adjoint. Part II: Retrieval Experiments of an Observed Florida Convective Storm. **Bulletin of the American Meteorological Society,** 1 March 1998, p. 835-852.

SUN J., N A CROOK. Real-Time Low-Level Wind and Temperature Analysis Using Single WSR-88D Data. **Bulletin of the American Meteorological Society,** 1 February 2001, p. 117-132.

**Titan Training Mendoza Province**, 2001, WMI Techonology Transfer and Research [Disponível na Biblioteca do IPMet – em Inglês].

WILSON, J W, N A CROOK, C K MUELLER, J SUN and M DIXON. Nowcasting Thunderstorms: A Status Report, **Bulletin of the American Meteorological Society,** 1998, v. 79 (10), p. 2079-2099.