ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995

Seria: INFORMATYKA z. 28 Nr kol. 1270

Piotr BAJERSKI

RÓWNOLEGŁA REALIZACJA OBLICZEŃ W SIECI KOMPUTEROWEJ W PROCESIE GENERACJI MAP ZANIECZYSZCZEŃ POWIETRZA*

Streszczenie. W pracy przedstawiono szkielet systemu wizualizacji przestrzennego rozkładu zanieczyszczeń powietrza, realizującego obliczenia w sposób rozproszony w sieci stacji roboczych Sun. Wyznaczony rozkład jest prezentowany na podkładzie mapy administracyjnej. Przedstawiono wyniki zastosowania systemu do wizualizacji zanieczyszczenia powietrza na terenie Aglomeracji Górnośląskiej wykorzystując dane uzyskane w sieci pomiarowej SANEPIDu.

DISTRIBUTED COMPUTING ON A COMPUTER NETWORK IN THE GENERATION OF AN AIR POLLUTION MAP

Summary. The paper presents a description of the framework of an air pollution spatial distribution visualization system. The system uses distributed computing on a Sun workstation network to speed the calculation up. The computed distribution is depicted on an administrative map. The results of using the system for visualization of air pollution distribution over the Upper Silesian Agglomeration, based on data collected in the SANEPID measurement network, are shown.

*Praca zrealizowana w ramach Projektu Badawczego (GRANTU) nr KBN 3 P406 011 04.

PARALLELE AUSFÜHRUNG DER BERECHNUNGEN IM COMPUTER-NETZ IM RAHMEN DES GENERIERUNGSPROZESSES DER LUFTVERSCHMUTZUNGSKARTEN

Zusammenfassung. Im Artikel wurde das Konzept eines Sichtbarmachungssystems räumlicher Luftverschmutzungsverteilung dargestellt. Das System führt verteilte Berechnungen im Netz der SUN-Workstations aus. Die berechnete Verteilung wird auf einer Verwaltungskarte vorgestellt. Der Artikel präsentiert Ergebnisse der Anwendung des Systems zur Sichtbarmachung der Luftverschmutzung im Bereich der Oberschlesischen Agglomeration und nützt dazu die im SANEPID-Vermessungsnetz erreichten Daten.

1. Wstęp

Znalezienie sposobów korzystania z zasobów środowiska bez jego dalszej degradacji, jak również minimalizowanie niekorzystnego oddziaływania istniejących zanieczyszczeń staje się coraz pilniejsze. Można powiedzieć, że jest wyzwaniem naszych czasów. Jednym z podstawowych warunków sprostania tym problemom jest dysponowanie możliwie pełnymi i wiarygodnymi informacjami o wielkości istniejących zanieczyszczeń i ich rozkładzie przestrzennym oraz prognozowanie rozwoju sytuacji. Informacje te są niezbędne do planowania inwestycji w ochronie środowiska oraz do podejmowania decyzji o lokalizacji nowych zakładów przemysłowych, osiedli mieszkaniowych lub obiektów poddanych szczególnej ochronie [8].

W województwie katowickim od końca lat sześćdziesiątych Wojewódzka Stacja Sanitarno-Epidemiologiczna prowadzi regularne pomiary w sieci, utworzonej przez kwadraty o bokach 8 km, zagęszczone w sieci jednostkowej do kwadratów 2 km x 2 km. W sieci tej znajduje się przeszło 800 stanowisk pomiarowych. Są one rozmieszczone z zagęszczeniem zależnym od zmienności obszarowej zanieczyszczeń powietrza, obejmując praktycznie obszar całego województwa (rys. 1). Z punktu widzenia dalszych rozważań ważne jest, że rozkład przestrzenny tych stanowisk odpowiada nieregularnemu, o zmiennej gęstości rozkładowi węzłów. W czerwcu 1993 roku oddano do eksploatacji automatyczną sieć monitoringu. W jej skład wchodzi 10 stacji automatycznych pomiarów imisji połączonych drogą telemetryczną z centralnym komputerem gromadzącym wyniki i przygotowującym prognozy. Do systemu dołączone są stacje meteorologiczne w celu uwzględnienia w analizie przemieszczania się zanieczyszczeń ruchów atmosfery [9].

Rys. 1. Węzły sieci pomiarowej SANEPIDu na terenie Aglomeracji Górnośląskiej
Fig. 1. Nodes of the SANEPID measurement network in the Upper Silesian Agglomeration

W procesie monitoringu zanieczyszczeń atmosfery gromadzone są duże ilości danych. Z punktu widzenia osoby podejmującej decyzje czy osoby zainteresowanej stanem środowiska ważne jest ich syntetyczne przedstawienie. Prezentację taką może zapewnić mapa zanieczyszczeń, na której przestrzenny rozkład zanieczyszczeń nałożony jest na mapę administracyjną. Ze względu na dużą ilość danych i konieczność wykonywania na nich czasochłonnych obliczeń problemy te wymagają dużej mocy obliczeniowej. Z grubsza rzecz ujmując, przyśpieszenie obliczeń można uzyskać zwiększając moc wykorzystywanego komputera lub zwiększając liczbę komputerów. Względy ekonomiczne przemawiają za drugim rozwiązaniem, w którym można wykorzystać posiadane komputery połączone siecią lokalną.

Niniejszy system realizowany był z myślą o tworzeniu map zanieczyszczeń w oparciu o dane pochodzące z sieci pomiarowej SANEPIDu. Chociaż prezentacja taka jest zagadnieniem dwuwymiarowym – dane są o postaci (x_i, y_i, F_i), $i = 1...N$, gdzie $P_i(x_i, y_i)$ reprezentuje współrzędne geograficzne węzła sieci pomiarowej, a F_i jest zmierzoną w nim wartością zanieczyszczenia – to system napisano na tyle ogólnie, że można go użyć do przetwarzania danych w przestrzeni trójwymiarowej. Możliwe jest to w dziedzinach,

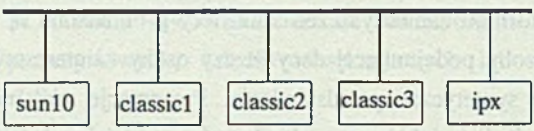
w których istnieją analogiczne problemy. Dla przykładu, można tu podać za [6] następujące zastosowania:

- modelowanie zmian pogody,
- wyznaczanie koncentracji zasobów naturalnych,
- badania ekonometryczne,
- modelowanie rozkładu temperatur w piecach hutniczych,
- obliczanie naprężeń w skrzydłach samolotów.

Metody interpolacji możemy podzielić na globalne i lokalne [5]. Globalne wykorzystują do wyznaczania wartości w danym punkcie wszystkie węzły znajdujące się na rozpatrywanym obszarze. Metody lokalne używają tylko węzłów leżących w pewnym ograniczonym sąsiedztwie tego punktu.

Użytkownik chcąc wygenerować mapę zanieczyszczeń określa rodzaj zanieczyszczenia, okres czasu (miesiąc i rok lub tylko rok), interesujący go obszar i przyporządkowuje przedziałom wartości zanieczyszczenia kolory.

2. Konfiguracja sprzętowa



Rys. 2. Konfiguracja sieci lokalnej
Fig. 2. The local network configuration

Eksperymenty były prowadzone na pięciu stacjach roboczych Sun (rys. 2), połączonych siecią Ethernet. Komputery te różniły się mocą obliczeniową. Najszybszy był komputer *sun10*, o architekturze SPARKstation 10, następny w kolejności był komputer *ipx*, o architekturze SPARKstation IPX, najwolniejsze były komputery *classic1*, *classic2*, *classic3* o architekturze SPARKclassic. Najszybszy komputer (*sun10*) był wykorzystany jako komputer nadzorujący obliczenia i obrazujący wyniki, ze względu na posiadanie przez niego kolorowego monitora i rezydowanie na nim serwera bazy danych. Pozostałe komputery pracowały jako komputery zdalne, realizując rozproszone obliczenia. Kompu-

tery pracowały pod kontrolą sunowskiej wersji UNIX-a i korzystały z sieciowego systemu plików NFS.

3. Narzędzia programowe

Program został stworzony jako aplikacja systemu OpenWindows, wykorzystując jego interfejs graficzny do prezentacji wyników — map zanieczyszczeń powietrza. Dane o zanieczyszczeniach w punktach pomiarowych, jak również dane dotyczące podziału administracyjnego województwa (przebiegu granic administracyjnych) i położenia punktów pomiarowych w terenie były przechowywane w bazach danych systemu Ingres. Kod programu został napisany w języku C. Do zarządzania synchronizacją między równolegle wykonującymi się procesami i wymiany danych między nimi został wykorzystany język C-Linda.

System Ingres jest systemem zarządzania rozproszoną, relacyjną bazą danych o architekturze klient—serwer [12]. W programie do pobierania danych z bazy danych wykorzystano język SQL zanurzony w języku C [13]. Instrukcje SQL w czasie prekompilacji są tłumaczone na serie funkcji wywołujących usługi serwera systemu zarządzania bazą danych, co ma znaczący wpływ na szybkość pobierania danych.

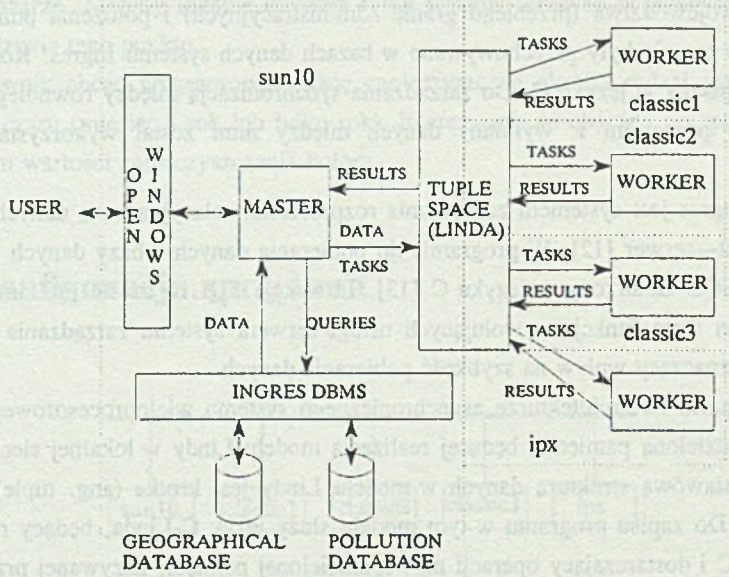
System oparto na architekturze asynchronicznego systemu wieloprocesorowego z wirtualnie współdzieloną pamięcią, będącej realizacją modelu Lindy w lokalnej sieci komputerowej. Podstawową strukturą danych w modelu Lindy jest krotka (ang. tuple), będąca ciągiem pól. Do zapisu programu w tym modelu służy język C-Linda, będący rozszerzeniem języka C i dostarczający operacji na współdzielonej pamięci, nazywanej przestrzenią krotek (ang. tuple space). Są to operacje:

- wysłanie krotki do przestrzeni krotek (operacja **out**),
- pobranie krotki z przestrzeni krotek (operacja **in**),
- odczytanie krotki z przestrzeni krotek (operacja **rd**),
- utworzenie aktywnej krotki, w ramach której tworzone są nowe, równolegle wykonujące się procesy (operacja **eval**).

Krotki są pobierane z przestrzeni krotek niedeterministycznie na podstawie dopasowania ich pól do wzorca (argumentu operacji *in* i *rd*). Operacje *in* i *rd* są blokujące. (Opis Lindy w języku angielskim można znaleźć w [1, 2, 16], języku polskim w [4, 14]).

4. Opis implementacji

Architektura systemu została przedstawiona na rys. 3. Program składa się z procesu nadzorcy i zmiennej liczby procesów wykonawców (na rysunku przedstawiono czterech wykonawców). Rys. 4 i 5 przedstawiają kod funkcji *master()*, z której tworzony jest proces nadzorcy, i funkcji *worker()*, odpowiadającej procesowi wykonawcy. W zapisie przyjęto pewne uproszczenia: nieistotny, z punktu widzenia zrozumienia idei programu, kod pominięto; część kodu zastąpiono komentarzem. Dla struktur danych tworzonych w programie dynamicznie przedstawiono tylko ich uproszczone deklaracje.



Rys. 3. Schemat systemu wizualizacji przestrzennego rozkładu zanieczyszczeń powietrza
Fig. 3. The scheme of the air pollution spatial distribution visualization system

Proces nadzorcy jest tworzony na komputerze *sun10* z funkcji *master()* po wybraniu przez użytkownika z menu aplikacji pozycji tworzenia mapy zanieczyszczeń. Proces nadzorcy prowadzi interakcje z użytkownikiem za pośrednictwem mechanizmów systemu Open Windows. Pobiera od niego parametry określające kontekst tworzonej mapy i na podstawie tego kontekstu odczytuje potrzebne dane z bazy danych geograficznych, a następnie z bazy danych z zanieczyszczeniami. W następnym etapie tworzy zdalne procesy wykonawców i generuje dla nich zadania umieszczając je w przestrzeni krotek. Pobierane

przez nadzorcę rezultaty są na bieżąco obrazowane na ekranie. Po zakończeniu obliczeń nadzorca wysyła "trujące pigułki", których "zażycie" kończy procesy wykonawców.

Proces wykonawcy w pętli: pobiera zadanie do wykonania, wyznacza podobszary, na których leżą węzły potrzebne do obliczeń, pobiera te węzły, oblicza wartości w punktach podobszaru leżących wewnątrz województwa i wyprowadza wynik. Pętla kończy się, gdy pobrane zadanie okaże się "trującą pigułką". W celu ograniczenia ilości przesyłanych danych proces wykonawcy odwzorowuje obliczoną wartość zanieczyszczenia w numer przedziału, w którym się on mieści.

```
master() {

        struct {
                int  xl, xr, yt, yb;            /* granice podobszaru */
                int  area_lenght;               /* wielkość podobszaru */
                char  area[];                   /* obszar do obliczeń */
        } tasks[];
        struct contour_ele  *contours;
        struct node  nodes[], *area_nodes[];
        char  result[];

        ReadMapContext(&context);               /* pobranie od użytkownika parametrów mapy */
        ReadContours(&context, &contours);      /* odczytanie danych  geograficznych */
        ReadPollution(&context, &nodes);        /* odczytanie zanieczyszczeń */

        start_timer();                          /* rozpoczęcie pomiaru czasu */
        /* utworzenie zadań */
        CreateTasks(&context, &contours, &nodes, &tasks, &area_nodes);

        for (i = 0; i < worker_count; i++)      /* utworzenie wykonawców */
                eval("worker", worker());
        for (i = 0; i < task_count; i++)        /* wysłanie zadań */
                out("task", i, task[i].xl, task[i].xr, task[i].yt, task[i].yb,
                task[i].area:task[i].area_lenght);
        for (i = 0; i < task_count; i++)        /* wysłanie węzłów */
                out("nodes", i, area_nodes[i]:);

        for (i = 0; i < task_count; i++) {
                /* pobranie i wyświetlenie wyniku */
                in("results", ?xl, ?xr, ?yt, ?yb, ? result:);
                ShowContours(xl, xr, yt, yb, result);
        };
        print_times();                          /* wypisanie czasu obliczeń */

        /* wysłanie "trujących pigułek" */
        for (i = 0; i < task_count; i++)
                out("task", -1, 0, 0, 0, 0, task[0].area:task[i].area_lenght);
}
```

Rys. 4. Kod procesu nadzorcy
Fig. 4. The master process code

Najprostszym podejściem do zrównoleglenia interpolacji jest podział obszaru na tyle podobszarów, ile komputerów bierze udział w obliczeniach i przydzielenie każdemu z tych komputerów fragmentu obszaru, na którym ma dokonać obliczeń. Nie zapewnia to jednak dynamicznego równoważenia obciążenia, które jest konieczne do uzyskania wysokiej efektywności, jeżeli poszczególne zadania różnią się złożonością, komputery są różnej mocy lub są w różnym stopniu obciążone innymi zadaniami.

```
worker() {

        char area[], result[][];

        while (1) {
                in("task", ? task_number, ?xl, ?xr, ?yt, ?yb, ? area:);
                if (task_number < 0)
                        break;                  /* trująca pigułka */
                Get_Nodes(task_number, &nodes)  /* pobranie potrzebnych węzłów */
                for (dla wszystkich puntów (x, y)
                                na podobszarze (xl, yt) (xr, yb))
                        if (punkt (x, y) leży na terenie województwa)
                                result[x][y] = Partition(Interpolation(x, y,
                                                nodes));
                out("result", xl, xr, yt, yb, result:);
        };
}
```

Rys. 5. Kod procesu wykonawcy
Fig. 5. A worker process code

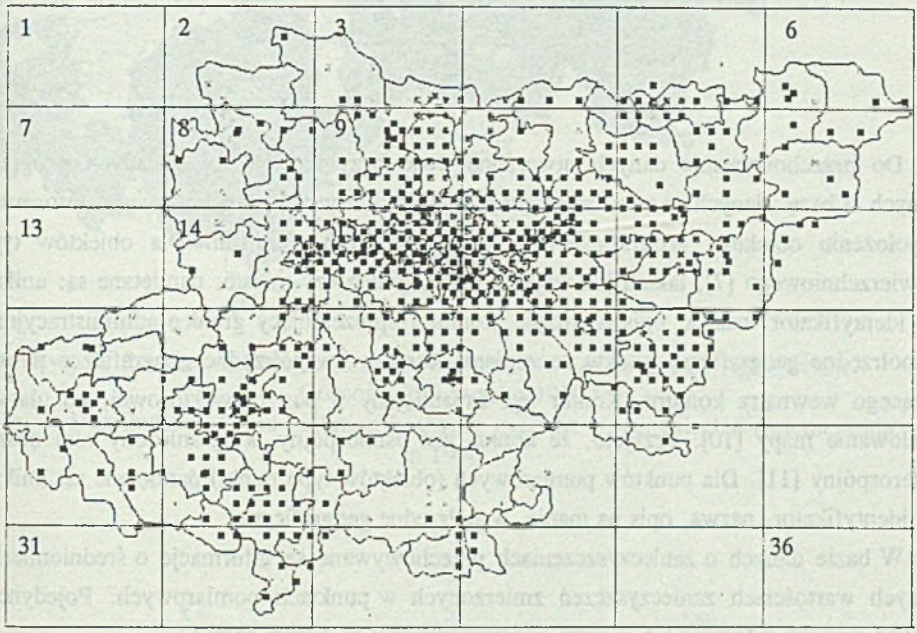
W rozważanym problemie dynamiczne równoważenie obciążenia jest ważne ze względu na różną złożoność poszczególnych zadań, jak i na możliwość obciążenia maszyn innymi obliczeniami. Co się tyczy zadań, to ich złożoność zależy od liczby punktów, dla których należy dokonać obliczeń, jak i od ilości węzłów na danym obszarze. Ilość punktów jest zmienna, ponieważ wartości są obliczane tylko w punktach leżących wewnątrz województwa.

Rozwiązanie oparto na schemacie nadzorca/wykonawca (ang. master/worker). Tworzymy w takim przypadku worek z zadaniami, tak że jednemu zadaniu odpowiada jedna krotka opisująca podobszar, na którym należy dokonać interpolacji. W jej polach zawarta jest wielkość podobszaru (wyrażona we współrzędnych ekranowych) i informacja określająca, które punkty leżą w granicach województwa.

Na rys. 6 przedstawiono przykładowy podział na podzadania. Każdy prostokąt symbolizuje podzadanie opisane krotką. Gdy dany proces pobierze zadanie do wykonania, potrzebne mu są wszystkie węzły leżące na tym podobszarze oraz węzły leżące poza tym podobszarem, a znajdujące się dostatecznie blisko punktów tworzących brzeg. "Dostateczną bliskość" można określić przyjmując pewien promień R, będący parametrem interpolacji.

Dla prostoty rozwiązania przyjęto, że węzły są rozdzielone na podobszary i w tej postaci znajdują się w przestrzeni krotek. Ponadto założono, co nie stanowi istotnego ograniczenia, że promień R nie przekracza wielkości podobszaru. Tak więc dany proces musi pobrać wszystkie podobszary przylegające do podobszaru, który interpoluje. Na przykład, jeśli wykonawca pobierze zadanie nr 8, to musi pobrać węzły z obszarów 1, 2, 3, 7, 8, 9, 13, 14, 15.

Z rys. 6 widać również, że pewne podzadania (np. 1, 36) nie wymagają żadnych obliczeń, gdyż podobszar, któremu odpowiadają, leży poza granicami województwa.



Rys. 6. Podział zadania na podzadania
Fig. 6. Splitting task into subtasks

Do testowania systemu użyto metody Sherpada – aproksymacji z wagą odwrotności odległości [6]. Metodę tę zmodyfikowano używając w obliczeniach tylko węzłów leżących w otoczeniu o promieniu R punktu, w którym wartość jest obliczana. W ten sposób otrzymano lokalną metodę interpolacji (wzory (1) i (2)). Promień R jest parametrem interpolacji, zadawanym przez użytkownika. $\|P - P_i\|$ oznacza odległość punktów P i P_i.

$$S(P) = \frac{\sum_{i=1}^{N} \frac{F_i}{\varrho_i^2(P)}}{\sum_{i=1}^{N} \frac{1}{\varrho_i^2(P)}} \tag{1}$$

gdzie

$$\frac{1}{\varrho_i(P)} = \frac{(R - \|P - P_i\|)_+}{\|P - P_i\|} \tag{2}$$

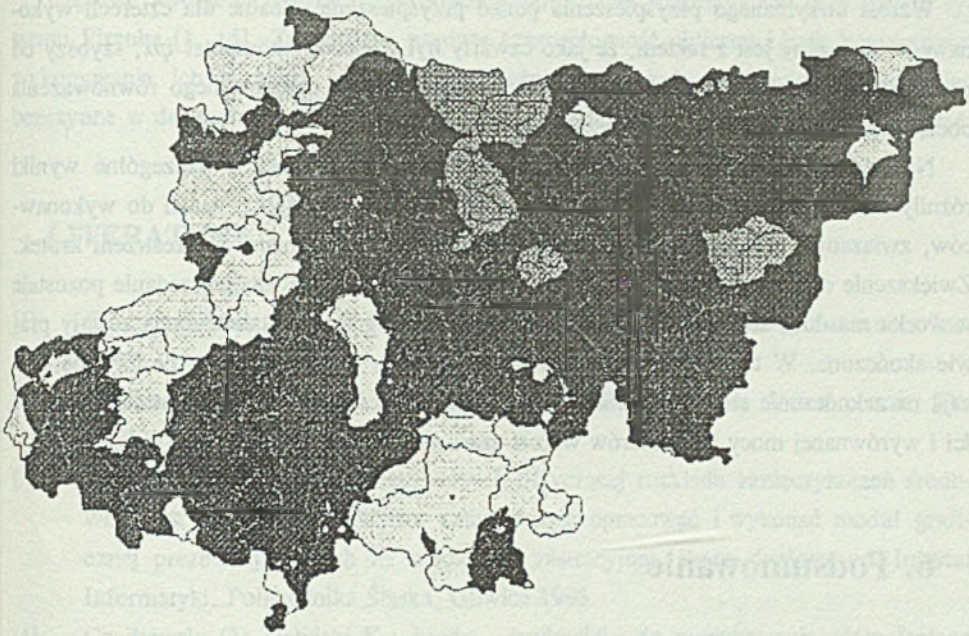
Do przechowywania danych utworzono dwie bazy danych: bazę danych geograficznych i bazę danych o zanieczyszczeniach [3]. Pierwsza przechowywała informacje o położeniu obiektów geograficznych i ich wzajemnych relacjach. Dla obiektów typu powierzchniowego [7], takich jak województwo, gmina czy miasto, pamiętane są: unikalny identyfikator, nazwa, opis na mapie, kontur (reprezentujący granicę administracyjną), współrzędne geograficzne punktu zaczepienia konturu i współrzędne geograficzne punktu leżącego wewnątrz konturu. Kontur jest zapamiętany w postaci wektorowej, co ułatwia skalowanie mapy [10]. Przyjęto, że kontur jest ośmiospójny, a ograniczony nim obszar czterospójny [11]. Dla punktów pomiarowych (obiektów typu punkt) pamiętane są: unikalny identyfikator, nazwa, opis na mapie, współrzędne geograficzne.

W bazie danych o zanieczyszczeniach przechowywane są informacje o średniomiesięcznych wartościach zanieczyszczeń zmierzonych w punktach pomiarowych. Pojedyncza tablica zawiera informacje o jednym zanieczyszczeniu w danym roku [3].

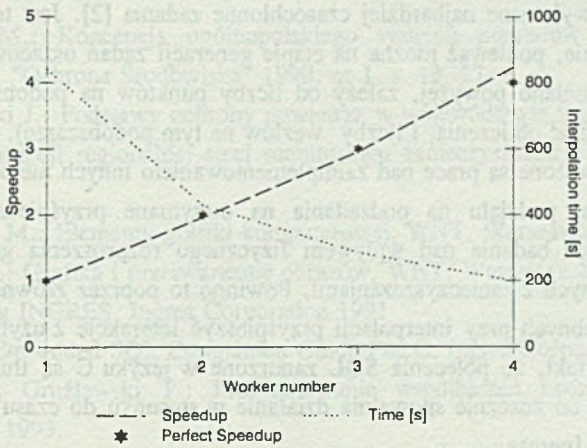
5. Otrzymane wyniki

Na rys. 7 przedstawiono wygenerowaną mapę zanieczyszczeń powietrza ołowiem na terenie Aglomeracji Górnośląskiej w roku 1990.

Wyniki przeprowadzonych eksperymentów są przedstawione na rys. 8. Eksperymenty były prowadzone na komputerach nie obciążonych innymi zadaniami. Otrzymanie przyśpieszenia bliskiego idealnemu jest związane z korzystnym stosunkiem czasu przesyłu danych do czasu obliczeń wykonywanych na tych danych. Ponadto rozwiązanie odznacza się dobrą skalowalnością.



Rys. 7. Rozkład zanieczyszczenia powietrza ołowiem na terenie Aglomeracji Górnośląskiej w roku 1990
Fig. 7. Distribution of lead pollution in the air over the Upper Silesian Aglomeration in 1990



Rys. 8. Otrzymane przyśpieszenie i czas interpolacji
Fig. 8. Achieved speedup and the time of interpolation

Wzrost otrzymanego przyśpieszenia ponad przyśpieszenie idealne dla czterech wykonawców związany jest z faktem, że jako czwarty był dołączony komputer *ipx*, szybszy od pozostałych. Obrazuje to skuteczność zastosowanej metody dynamicznego równoważenia obciążenia.

Na wykresie zostały przedstawione średnie czasy wykonania. Poszczególne wyniki różniły się między sobą, co było powodowane różnym przydziałem zadań do wykonawców, związanym z niedeterminizmem pobierania krotek z zadaniami z przestrzeni krotek. Zwiększenie czasu wykonania występowało w sytuacji, w której ostatnie zadanie pozostałe w worku ma dużą złożoność i pobierane jest w chwili, gdy pozostałe zadania zostały prawie skończone. W takiej sytuacji wszystkie komputery, oprócz jednego, bezczynnie czekają na zakończenie się tego zadania. Dzięki dużej ilości zadań o zróżnicowanej złożoności i wyrównanej mocy komputerów wzrost czasu nie przekraczał dziesięciu procent.

6. Podsumowanie

Zaproponowany sposób rozpraszania obliczeń w sieci komputerowej zapewnia dużą efektywność wykorzystania komputerów. Otrzymane wyniki sugerują, że rozwiązanie odznacza się dobrą skalowalnością, tzn. zwiększanie liczby komputerów powodowałoby proporcjonalny przyrost przyśpieszenia, dopóki podzadania byłyby dostatecznie złożone i odpowiednia ich liczba przypadała na każdy z komputerów.

Problem wzrostu czasu wykonania związany z niekorzystnym przydziałem zadań do wykonawców można rozwiązać poprzez uporządkowanie zadań w ten sposób, że jako pierwsze byłyby wykonane najbardziej czasochłonne zadania [2]. Jest to możliwe w opisywanym problemie, ponieważ można na etapie generacji zadań oszacować ich złożoność (która, jak wspomniano powyżej, zależy od liczby punktów na podobszarze, w których należy przeprowadzić obliczenia, i liczby węzłów na tym podobszarze).

Obecnie prowadzone są prace nad zaimplementowaniem innych metod interpolacji, jak również wpływem podziału na podzadania na otrzymane przyśpieszenie interpolacji. W toku są również badania nad wpływem fizycznego rozproszenia geograficznej bazy danych i bazy danych z zanieczyszczeniami. Powinno to poprzez zrównoleglenie pobierania danych potrzebnych przy interpolacji przyśpieszyć interakcję z użytkownikiem. Duże znaczenie ma tu fakt, że polecenia SQL zanurzone w języku C są tłumaczone na ciągi wywołań funkcji, co znacznie spowalnia działanie w stosunku do czasu wykonania zapytań w środowisku Ingresa.

Ciekawe wydaje się również zastosowanie paralelizmu adaptacyjnego przy użyciu systemu Piranha [1, 15]. Ze względu na dużą czasochłonność obliczeń i brak konieczności wykonywania ich w czasie rzeczywistym można generować serie map wykorzystując bezczynne w danej chwili komputery, wycofując obliczenia, gdy ktoś zacznie ich używać.

LITERATURA

[1] C-Linda User's Guide & Reference Manual, Scientific Computing Associates Inc. 1993.

[2] Carriero N., Gelernter D.: How to write parallel programs, A first course. The MIT Press 1990.

[3] Bajerski P.: Dla systemu bazy danych dotyczącej rozkładu zanieczyszczeń środowiska na terenie województwa katowickiego opracować i wykonać moduł graficznej prezentacji danych na mapie administracyjnej. Praca dyplomowa. Instytut Informatyki. Politechnika Śląska, Gliwice 1993.

[4] Czajkowski G, Zieliński K.: Linda - środowisko do przetwarzania równoległego i rozproszonego w sieciach stacji roboczych. Informatyka nr 5, 1993.

[5] Sabin M.: Contouring - the State of the Art, "Fundamental Algorithms for Computer Graphics", Springer-Verlang Berlin 1985.

[6] Nielson M.: Scattered Data Modeling, "IEEE Computer Graphics & Applications", Vol 1, 1993.

[7] Werner P.: Wprowadzenie do geograficznych systemów informacyjnych. Warszawa 1992.

[8] Nowicki M.: Koncepcja ogólnopolskiego systemu pomiarów zanieczyszczenia atmosfery, "Ochrona Środowiska" 1990, nr 1, s. 13 -27.

[9] Walczewski J.: Podstawy ochrony powietrza w województwie katowickim, opracowanie koncepcji regionalnej sieci monitoringu zanieczyszczeń powietrza. Kraków 1993.

[10] Jankowski M.: Elementy grafiki komputerowej, WNT, Warszawa 1990.

[11] Pavlidis T.: Grafika i przetwarzanie obrazów, WNT, Warszawa 1987.

[12] Introducing INGRES. Ingres Corporation 1991

[13] INGRES/Embedded SQL Companion Guide For C. Ingres Corporation 1991.

[14] Weiss Z., Gruźlewski T.: Programowanie współbieżne i rozproszone. WNT, Warszawa 1993.

[15] Carriero N., Freeman E.,Gelernter D.: Adaptive Parallelism on Multiprocessors: Preliminary Experience with Piranha on the CM-5. Department of Computer Science, Yale University 1993.

[16] Bjornson R.: Linda on Distributed Memory Multiprocesors. Dissertation. Department of Computer Science, Yale University 1993.

Recenzent: Dr inż. Andrzej Wilk

Wpłynęło do Redakcji 23 listopada 1994 r.

Abstract

The paper presents a description of the framework of an air pollution spatial distribution visualization system. The system was implemented on a Sun workstation network using C-Linda language for coordination programming and Ingres DBMS for pollution and geographical data managing. Both databases are also described.

The master/worker approach was used to split the task into subtasks and achieve dynamic work load balancing.

The system was tested using data gathered in the SANEPID measurement network covering the Upper Silesian Agglomeration. The network consists of over 800 nodes placed irregularly over the area.

The results show that the air pollution map generation problem can be solved using the Linda model on a local network achieving good performance (fig. 8). Ways of tuning the application to attain even better speedup are suggested.